The Atomic Lambda-Mu Calculus

He, Fanny

Award date:
2018

Awarding institution:
University of Bath

The Atomic Lambda-Mu Calculus

submitted by

Fanny He

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Computer Science

January 2018

Signature of Author .....

Fanny He

Abstract

A cornerstone of theoretical computer science is the Curry-Howard correspondence where formulas are types, proofs are programs, and proof normalization is computation. In this framework we introduce the atomic $\lambda\mu$-calculus, an interpretation of a classical deep inference proof system. It is based on two extensions of the λ-calculus, the $\lambda\mu$-calculus and the atomic λ-calculus. The former interprets classical logic, featuring continuation-like constructs, while the latter interprets intuitionistic deep inference, featuring explicit sharing operators.

The main property of the atomic λ-calculus is reduction on individual constructors, derived from atomicity in deep inference. We thus work on open deduction, a deep inference formalism, allowing composition with connectives and with derivations, and using the medial rule to obtain atomicity. One challenge is to find a suitable formulation for deriving a computational interpretation of classical natural deduction. A second design challenge leads us to work on a variant of the $\lambda\mu$-calculus, the $\Lambda\mu S$-calculus, adding streams and dropping names.

We show that our calculus has preservation of strong normalization (PSN), confluence, fully-lazy sharing, and subject reduction in the typed case. There are two challenges with PSN. First, we need to show that sharing reductions strongly normalize, underlining that only β, μ-reductions create divergence. Our proof is new and follows a graphical approach to terms close to the idea of sharing. Second, infinite reductions of the atomic calculus can appear in weakenings, creating infinite atomic paths corresponding to finite $\Lambda\mu S$-paths. Our solution is to separate the proof into two parts, isolating the problem of sharing from that of weakening. We first translate into an intermediate weakening calculus, which unfolds shared terms while keeping weakened ones, and preserves infinite reductions. We then design a reduction strategy preventing infinite paths from falling into weakenings.

Acknowledgements

I would like to thank my supervisor Willem Heijltjes for guiding me through this PhD, I couldn't have done this without his insight and support, and no sentence could fully express my gratitude to him.

I am also deeply grateful for insightful discussions with my second supervisor Guy McCusker, who was also my first contact in Bath.

I would also like to thank my examiners Alessio Guglielmi and Damiano Mazza for taking the time to offer their feedback on my dissertation, thank you for your comprehensive and useful remarks.

To Paul-André Melliès, Michele Pagani, and Alexis Saurin, you gave me the courage for this adventure, thank you for your kindness and encouragement.

Thanks to Valentin Blot, Etienne Duchesne, Twey Kay, Dave Sherratt for many captivating discussions about work and life.

I also greatly benefited from discussions with Anupam Das, Delia Kesner, and Jim Laird.

For their encouragement and kindness, I would like to thank Pierre Boudes, Paola Bruscoli, Giulio Manzonetto, and Lionel Vaux.

Thanks to the friends and colleagues I met in Bath and elsewhere, who were there when times were hard, who could uplift me with their conversations and inspire me. You've all been a great help.

For their continued support and love, I would also like to thank my parents. Thank you for giving me a push forward when I needed it most.

Nans, I would be a completely different person if I hadn't met you. For letting me rely on you, for your infallible support and trust, thank you.

Contents

Chapter 1

Introduction

In this thesis, we investigate the atomic $\lambda\mu$-calculus, a computational calculus corresponding to classical logic inspired by the deep inference methodology. This chapter first skims through the history of the questions that shaped proof theory, its methodology and aims, underlying our work. This intellectual tradition shapes the research questions investigated in the next chapters. We then introduce some of the more recent technical progress on which our research builds, the classical version of the Curry-Howard correspondence, and the deep inference formalism. Another section describes some motivations behind our work. The chapter ends with an outline of the thesis.

1.1 Early to modern proof theory

In this section, we first describe the birth of proof theory. Then, we give a short account of its influence on the mathematician's approach to (the philosophy of) mathematics. It concludes by briefly linking historical developments to modern trends in proof theory.

Proof theory has its roots in the foundational crisis of mathematics. Mathematics and logic were constantly subject to "language paradoxes", such as the Cretan liar stating "I am lying (right now)". Whilst this sentence could be dismissed as a joke, the absence of formal context could not provide any satisfying explanation for why this is not a mathematical statement, or why one could not formalize such a contradiction in the mathematics of arithmetic or analysis. Unfortunately, with insufficiently defined notions such as functions or continuity, paradoxes did appear, and even worse, attempts to formalize mathematics were often themselves subject to contradictions (thus "naive" set theory). Were mathematics fundamentally flawed, unable to provide a context to differentiate truth from falsity? Hilbert thus formulated a plan for a solution, which

became known as Hilbert's program. The idea was to give a simple axiomatic system, allowing only (meta-mathematical) finitist means, that could be used to demonstrate all theorems, while being safe from any inconsistency. The finiteness would ensure that correctness can be proved in a purely mechanically verifiable way, purging any doubt of absolute truth. This program can be seen as a strong manifestation of the positivist philosophy of the post-enlightenment era, positing that through the sole pursuit of scientific reason humanity will solve all problems in a definite way. Unfortunately, Gödel's theorems came as the demise of this program, as they show that no such system could exist. No system strong enough to encode arithmetic can be complete, consistent, and decidable. Yet Gentzen gave a proof of arithmetic's consistency through the cut-elimination of sequent calculus. Since it cannot contradict Gödel's theorems, this proof had to abandon parts of Hilbert's initial plan, and indeed it only gives a proof of relative consistency, that cannot be established without appeal to the consistency of a stronger system. The introduction of his proof systems were instrumental in establishing proof theory, since beyond the syntactical game of proofs it introduced new ways to see mathematics and showed how it could lead to proofs of substantially nontrivial theorems. Proof theory, which had been forming in the works of logicians such as Frege and Russell, is thus widely seen as coming out of the ruins of Hilbert's program.

The foundational crisis, besides its philosophical inquiry into the ontology of mathematics, thus also embodied an absolutist view of science, which was contested by eminent mathematicians. This lead to acrimonious feuds between mathematicians, and in particular between Hilbert and Brouwer, who discarded formalism as a mere symbolic game, and promoted the intuitionistic school of mathematics. The logical approach continued to be seen as unfruitful for true mathematics for a long time, with René Thom going as far as saying "Whatever is rigorous is insignificant". In other words, the price of formalism is so high that it prevents from doing any "true mathematics" while being rigorous. The success of formalism, ultimately, was really foreign to the mathematician's activity. Poincaré, on the subject, famously said "Logic sometimes makes monsters. For half a century we have seen a mass of bizarre functions which appear to be forced to resemble as little as possible honest functions which serve some purpose. More of continuity, or less of continuity, more derivatives, and so forth. [...] In former times when one invented a new function it was for a practical purpose; today one invents them purposely to show up defects in the reasoning of our fathers and one will deduce from them only that." His position on the importance of the formalist approach, or rather the duality and complementarity of the intuitionistic and the formalist nature of the mathematician's work, was beautifully presented in his talk

"De l'intuition et de la logique en mathématiques". This would have been the definite opinion on the intuitionist/formalist approach if it wasn't for the convergence of the two, due to the Curry-Howard-Lambek correspondence, or computational trinitarianism. This gives a new dream of seamlessly doing high-level creative mathematics and proving them formally, as heralded by Voevodsky. While this is outside of the scope of this thesis, his program of univalent foundations of mathematics is a complete reenvisioning of the foundations in a proof-theoretic light, through homotopy type theory (HoTT) [Uni13].

The striking foresight of Hilbert's program lies in how it anticipated the digital age, for the use of a formal language to encode mathematics opens the possibility to check proofs completely automatically. Beyond that, the work on computers led people to investigate the nature and meaning of computation, formalized by Turing machines and the λ-calculus. Surprisingly enough, the two formalisms are equivalently powerful, each one can encode the other, thus encode any computation (the Church-Turing hypothesis). But even more surprising was that the λ-calculus is equivalent to natural deduction, the intuitionistic logical system; far from being a mere mathematical curiosity, this result shows that computing is proving, and proving is computing. This realization was then enriched by the equivalence of the equational theory with cartesian closed categories. These equivalences were extended in many ways, and most prominently to classical logic, giving an equivalence between control operators and Peirce's law. This formal equivalence is also extended to an informal version used to guide intuitions about general purpose programming languages primitives and their mathematical power. By bridging the gap between high-level programming languages and formal methods, it is thus possible to express complex mathematical ideas in a natural way, to prove the correctness of programs, and to program proofs of correctness. Great advances have been made on these fronts since De Bruijn's theorem checker Automath, forerunner of proof assistants, that are currently spearheaded by Racket, Scala and Haskell in programming language research and Agda and Coq as proof assistants.

1.2 Technical introduction

In this section, we introduce two independent mathematical developments, proof systems and computational calculi, which were later proved to be equivalent. Since our work builds on the interplay between the two, we first introduce two simple systems and describe the equivalence. We then introduce two parallel works extending the first systems, giving more involved correspondences. Our calculus is directly constructed to

cast them in a unified framework. In the first subsection, we introduce natural deduction, the proof system used in this thesis. The second section introduces the λ-calculus, a simple model of computation. The third subsection describes the link between these two domains, or Curry-Howard correspondence. It is followed by a brief introduction to category theory, which is then linked as the third aspect of computational trinitarianism. The fourth section gives the correspondence in the deep inference setting, extending the λ-calculus with the atomicity property. The fifth section presents the classical Curry-Howard correspondence, extending to classical natural deduction and introducing a new operator to the calculus.

1.2.1 Proof theory

Proof theory aims, as its name suggests, to give a mathematical framework to reason on proofs as mathematical objects, on which one may reason and deduce properties. It was a key element of Hilbert's program, that introduced *Hilbert-style* proof systems with one main inference rule, modus ponens, which from a proof of A and a proof of $A \to B$ gives a proof of B. Modus ponens was seen as epitomizing direct reasoning in mathematics, and thus was given the central role in Hilbert's proof theory. When Gentzen introduced his proof systems, *natural deduction* and *sequent calculus* [Gen35, Gen69], in which the modus ponens is represented by the *cut rule*, his proof of consistency relied on cut-elimination. By removing all cuts, we get a normal proof that has the subformula property, i.e. every formula that appears in the reasoning must appear in the conclusion, and this way one cannot complete a proof of falsity. In other words, modus ponens, the central rule of Hilbert systems, can be completely dispensed with. But the cut rule is not a superfluous rule, as the equivalent cut-free proof can be much bigger than the initial proof. A proof with a cut thus is an implicit object, that can be made explicit through cut-elimination; in Girard's metaphor, it is like a cheque, that you can treat as an explicit object (if this is a cheque by Donald Knuth, you frame it on your wall), or you can treat it as an implicit object, go to the bank and cash it.

In this thesis, we work on natural deduction systems, that we will present with sequents to underline the link with term calculi. Let Latin letters $A, B, \ldots$ denote formulas, and Greek letters Γ, Δ denote lists of formulas. Having a *sequent* (or *judgment*) $\Gamma \vdash \Delta$ means that whenever all elements in the list of *premises* (or *assumptions*) Γ are true, one formula in the *conclusion* Δ is true. To obtain a proof system, we must now be able to apply rules to act on proofs; these are written in a tree form:

$$\frac{\Gamma_1 \vdash \Delta_1}{\Gamma_2 \vdash \Delta_2}\,\mathtt{r}$$

In other words, it means that by applying an inference rule $\mathtt{r}$, we obtain a proof (or *derivation*) of $\Gamma_2 \vdash \Delta_2$ from a proof of $\Gamma_1 \vdash \Delta_1$. So there is a notion of natural, or meta implication (the rule) and the implication in proofs ($A \to B$).

In Gentzen's original natural deduction, sequents consist of a list of formulas Γ as premise, and of one formula C as conclusion. Rules work on the conclusions on the right hand side of $\vdash$, and can either *introduce* connectives, or dually *eliminate* them. The system (in propositional logic) is built with the following rules:

$$\frac{}{C \vdash C}\,\mathtt{Ax} \qquad \frac{\Gamma, A \vdash C}{\Gamma \vdash A \to C} \to_{\mathtt{i}} \qquad \frac{\Gamma \vdash A \to C \qquad \Gamma' \vdash A}{\Gamma, \Gamma' \vdash C} \to_{\mathtt{e}} (\mathtt{cut})$$

$$\frac{\Gamma \vdash C_1 \qquad \Gamma' \vdash C_2}{\Gamma, \Gamma' \vdash C_1 \wedge C_2} \wedge_{\mathtt{i}} \qquad \frac{\Gamma \vdash C_1 \wedge C_2}{\Gamma \vdash C_j} \wedge^j_{\mathtt{e}}$$

$$\frac{\Gamma \vdash C_j}{\Gamma \vdash C_1 \vee C_2} \vee^j_{\mathtt{i}} \qquad \frac{\Gamma \vdash C_1 \vee C_2 \qquad \Gamma', C_1 \vdash C_3 \qquad \Gamma'', C_2 \vdash C_3}{\Gamma, \Gamma', \Gamma'' \vdash C_3} \vee_{\mathtt{e}}$$

If we restrict this system to the three first rules, we obtain a system corresponding to minimal logic, the fragment obtained by restricting formulas to axioms and implication rules.

From there, many variant formalisms were developed with different motivations such as a close relation to a computational calculus, or good complexity properties (such as the deep inference formalism studied later).

1.2.2 The λ-calculus

Church's own approach to the foundation of mathematics led him to introduce the λ-calculus [Chu32], a model of computation which can be seen as a formal functional programming language. It is a theory whose objects are functions (called terms), and whose subject is the function application procedure (called β-reduction).

Terms of the λ-calculus are defined by the following syntax:

$$t, u ::= x \mid \lambda x.t \mid (t)\, u$$

where x is a variable, $\lambda x.t$ is a λ-abstraction (or an anonymous function), and $(t)\, u$ is the application of the term t to the term u. The main relation between terms is

β-reduction, which corresponds to function application. Applying the λ-abstraction $\lambda x.t$ to the argument u gives t in which all occurrences of x have been replaced by u:

$$(\lambda x.t)u \longrightarrow_\beta t\{u/x\}$$

In other words, the term is reduced by applying a reduction step, or equivalently a reduction corresponds to one step of a computation. A term is in normal form if no reduction can be applied, and thus corresponds to the result of a computation. The notation $t_1 \longrightarrow_{\beta^*} t_2$ denotes that t_2 can be obtained by an arbitrarily long sequence of reductions (it is the reflexive transitive closure of the β-reduction).

With this extremely terse syntax, it is not obvious to see that in fact, every computable function can be written as a λ-term. To show it, Church gave an encoding of arithmetic. In particular, he represented natural numbers in the λ-calculus, using what is now known as the Church integers:

$$\overline{n} \equiv \lambda f.\lambda x.\underbrace{(f)\ldots(f)}_{n}x.$$

To define the arithmetic operations, one needs to define terms that act on Church integers like their counterparts act on integers; for example the addition operator can be written

$$\overline{+} \equiv \lambda m.\lambda n.\lambda f.\lambda x.(m)f((n)fx)$$

and one can thus check that $n + m \equiv (\overline{+})\overline{n}\,\overline{m} \longrightarrow_{\beta^*} \overline{n+m}$.

One operation, the predecessor, was notably difficult to encode, and Church was convinced of the impossibility to find a λ-term for it until his student Kleene found an answer, at the dentist, while getting his wisdom teeth pulled.

However, the λ-calculus, seen as a logical system, is not consistent; one can encode the dreaded Russell paradox by the term $\Omega \triangleq (\lambda x.(x)x)\lambda x.(x)x$, which infinitely β-reduces to itself. Therefore Ω has no normal form, and some terms correspond to computations that never terminate. Yet there is a notion of algorithmic consistency that can be retrieved, the *confluence* property of the calculus. If there is a normal form, it is unique, and there is always a finite sequence of reductions that leads to the normal form. Rome does not always exist, but if it does, then there is a way to reach it. The

confluence property can be illustrated by the following diagram.

$$\begin{array}{ccc} u & \xrightarrow{\beta^*} & t_1 \\ \downarrow{\scriptstyle \beta^*} & & \downarrow{\scriptstyle \beta^*} \\ t_2 & \xrightarrow{\beta^*} & v \end{array}$$

For any term u, if there exist two reduction paths to terms t_1 and t_2, then there is a way to close the "diamond" diagram by finding a term v such that there exist two reduction paths from t_1 and t_2 to v.

In order to forbid infinite computations and this way get a logically consistent system, we can assign types to terms, in the spirit of Russell and Whitehead's Principia Mathematica, that introduced type theory as a solution to Russell's paradox. In this setting, each term is assigned a type, and the formation of the term must be consistent with the term types. For instance, the simply typed λ-calculus is obtained with only one type constructor, the function ($\rightarrow$) constructor:

$$A, B ::= a \mid A \rightarrow B$$

where a is a base (atomic) type. In particular, an abstraction $\lambda x.t$ constructed with a variable x of type A and a term t of type B (in which all free occurrences of x are of type A) will be given the type $A \rightarrow B$, and an application $(t)u$ can be typed if t has a function type $A \rightarrow B$ and u has the type A (the argument type of t). Terms in this setting not only always have a normal form, but any sequence of reduction leads to the normal form, a result known as strong normalization.

Furthermore, it is possible to omit types entirely, and have the compiler infer the correct types when they exist. The Hindley-Milner algorithm performs type inference in a very efficient way for the λ-calculus with some parametric polymorphism added through the `let` construct, and forms the base of type inference for many functional programming languages. The simply typed system is as its name suggests very simple, but it does not allow us to write natural abstractions, as every type needs its own identity function. Type polymorphism allows us to write $\forall\alpha.\alpha \rightarrow \alpha$ as the type of the identity function that works for any type. With this type system, it is possible to program with natural abstractions, such as lists or trees, that are called in general purpose programming languages as generic data structures.

In simply typed systems terms only depend on terms, but it is possible to consider calculi with terms depending on types (which gives polymorphism), types depending on types (type operators) and types depending on terms (dependent types). By consider-

ing each notion of type constructors as a dimension, we obtain the λ-cube, whose origin is the simply typed λ-calculus, and whose furthest corner corresponds to the calculus of constructions (CoC), the type system underlying Coq. Each extension can be added while retaining strong normalization, but proving so is a hard task. One of the most important breakthroughs is Girard's proof through the *candidates of reducibility* technique of the strong normalization of System F [Gir72], or λ-calculus with parametric polymorphism. Since proving the normalization of first-order typed calculi corresponds to the consistency of arithmetic, or equivalently the consistency of the ordinal ε_0, the normalization of second-order typed calculi corresponds to second-order arithmetic, in other words of *analysis*. However the extended expressive power of System F compared to the first-order polymorphism of Hindley-Milner comes at the price of type inference's decidability, in other words the compiler needs type annotations to perform its task.

1.2.3 The Curry-Howard-Lambek correspondence

The Curry-Howard-Lambek correspondence describes the tight connection between logical systems, computational calculi, and category theory. Thus, there are three lenses through which we can see computation, as in the old tale where blind men each give their own description of an elephant after touching it, but all descriptions seem incompatible. In fact, each one describes a different part (the trunk, legs and tail) without understanding the whole animal. The Curry-Howard-Lambek correspondence is the way to synthesize their views in a coherent sketch. This correspondence has a formal aspect, where the correspondence is tight (an isomorphism) and allows us to directly transfer results between the formalisms, and a looser aspect primarily used to guide intuitions.

The rigorous aspect was discovered first by Curry, through the equivalence between combinatory logic and Hilbert systems [Cur34]. The correspondence between λ-calculus and natural deduction was not to be made until several decades later by Howard [How80]. Basically, one obtains a proof in minimal logic by erasing the term in a typing judgment in simply typed λ-calculus:

$$\frac{}{x : A \vdash A}\ \text{Var} \qquad \frac{t : \Gamma, A \vdash B}{\lambda x.t : \Gamma \vdash A \to B}\ \lambda$$

$$\frac{t : \Gamma \vdash A \to B \qquad u : \Gamma' \vdash A}{(t)u : \Gamma, \Gamma' \vdash B}\ @$$

This correspondence was then extended to a trinity when Lambek explicitly described the link between the equational theory of simply typed λ-calculus and cartesian closed

categories (CCC's). This approach prompted the study of categorical semantic models of programming languages.

Eilenberg and MacLane's category theory is in a sense "a mathematics of mathematics", as it studies mathematical theories as *objects* and *morphisms* (transformations) preserving their structure. The only requirement is that the composition of morphisms is always defined, and that composition is associative. Since these requirements are very lax, many theories can be studied this way, such as sets with functions, algebraic structures with homomorphisms, or sets with preorder relations. The study thus focuses on the interaction between objects (their "social life"), which forms a specific class of category (e.g. CCC's). Since any structural invariant on a class of categories can be applied to the corresponding mathematical structure, these are very general results, to the point of being facetiously called "general abstract nonsense". A *functor* is then a transformation between categories (a morphism in the category of categories), that maps objects in the first category to objects in the second category, and morphisms to morphisms, consistently with the composition requirement. If categories are 0-dimensional objects (points), functors are thus 1-dimensional (lines), and one gets 2-dimensional objects by considering *natural transformations*, or morphisms between functors (in a 2-category). This concept is the main motivation behind category theory, as it captures in a mathematical notion the vague mathematician's intuition that some structures are naturally related. Note that in category theory we can simply reverse the arrows to get the dual category, and thus essentially get the dual notions for free (the initial object is dual to the terminal object, a product is dual to the coproduct).

The prototypical category is **Set**, with sets as objects and functions as morphisms. For any sets A and B, its cartesian product $A \times B$ is a set, therefore **Set** should include all finite cartesian *products* of sets, the empty product $\top$ being called the *terminal object*. An example of functor F from **Set** to itself is the product by another set X, taking any set A to the set $A \times X$. Furthermore, since for any sets A and B, functions from A to B form a set, there should be an object $A \Rightarrow B$ representing the set $\mathbf{Set}(A, B)$ of morphisms from A to B. This object is called the *exponential object*, and is defined satisfying the main condition:

$$\mathbf{Set}((C \times A), B) \cong \mathbf{Set}(C, (A \Rightarrow B))$$

It states that given any set C, we obtain an isomorphic natural transformation between $\mathbf{Set}((C \times A), B)$ and $\mathbf{Set}(C, (A \Rightarrow B))$, a result know as *currying*, which intuitively states that a function from a product $C \times A$ to B is equivalent to a function from C that returns a function from A to B. A category where all finite products and exponential

objects are defined is called a cartesian closed category. In a sense, CCC's thus capture the categories, such as **Set**, that are functionally well-behaved.

The idea behind the categorical account of the correspondence is to identify the equational theory of the logic system with the internal language of a specific class of categories. To achieve that, the functional nature of terms of the λ-calculus naturally suggests to represent terms as morphisms, with the composition being the application. The objects are thus the types, and the set of terms of type $A \to B$ in a category $\mathcal{C}$ simply becomes $\mathcal{C}(A, B)$, the set of morphisms from A to B. The identity terms $(\lambda x.x)$ naturally play the role of identity morphisms for each type. To encode the abstraction constructor, intuitively, we want to have "morphisms" taking "morphisms" as inputs. This describes the notion of exponential object. It follows that the equational theory of the simply typed λ-calculus describes the structure of CCC's.

Thus the Curry-Howard-Lambek correspondence informs us, like Monsieur Jourdain who did not know he was speaking prose all along, that computing is proving, and proving is computing. This revelation gave a new way to envision correct programs: instead of debugging and testing, one proves that it computes the correct result. De Bruijn implemented this aspect of the correspondence in Automath, introducing a formal language designed to allow automatic proof-checking for mathematical theories, which paved the way for modern proof assistants. It is worth noting that De Bruijn was unaware of Howard's work, and thus stated the correspondence independently, as well as introducing ideas that were later reinvented or became part of proof theory, such as dependent types, or technical innovations such as explicit substitutions and De Bruijn indices.

The three-way correspondence can be informally summarized in the following chart:

Programming	Logic	Categories
Type	Formula	Object
Typed program	Derivation	Morphism
Unit type	True	Terminal object $\top$
Void type	False	Initial object $\bot$
Function composition	Cut	Morphism composition
Product type	Conjunction	Product functor
Disjoint sum type	Disjunction	Coproduct functor
Function type	Implication ($\to$)	Exponential functor

1.2.4 Deep inference and the atomic λ-calculus

Deep inference was introduced as a general methodology to introduce proof systems with better complexity-theoretic properties, less bureaucracy, and a simpler syntax. Its central concepts are linearity, i.e. the quantification of resource usage, and geometry, i.e. the locality of information.

The methodology of deep inference was introduced to express logics which are not definable in Gentzen's sequent calculus, such as Guglielmi's BV [Gug07]. The "deep" qualifier comes from the fact that rules can be applied at any depth of a formula, in opposition to "shallow" inference where rules can solely be applied to the root of a formula (seen as a tree). In deep inference all rules are *local*, i.e. they can be checked in constant time, and *atomic* i.e. their application can be restricted to atoms. The first deep inference formalism is the *calculus of structures*, introduced as a deduction system enjoying top-down symmetry, i.e. proofs can be negated and flipped upside down, retrieving the duality between identity and cut that was concealed in the tree derivations of sequent calculus. However, because this system is sequential, i.e. proofs are made of sequences of formulas to which rules are applied, it differentiates some proofs that are logically equivalent, such as:

$$\dfrac{\dfrac{A \wedge B}{A \wedge D}}{C \wedge D} \neq \dfrac{\dfrac{A \wedge B}{C \wedge B}}{C \wedge D}$$

Open deduction was introduced later as a more general formalism, able to obtain this symmetry by giving proofs a two dimensional structure. For instance, the derivations above are identified as:

$$\frac{A}{C} \wedge \frac{B}{D}$$

Note that deep inference is a general formalism that can be widely applied. Deep inference systems have been developed for classical [BT01], intuitionistic [Tiu06], linear [Str02], and some modal logics [SS05], still satisfying the same proof-theoretic properties as in traditional systems such as cut-elimination [Brü06, Brü03a] or more generally *normalization*, giving a notion of normal form for proofs. Using *atomic flows* i.e. graphs tracing structural rules of a proof, normalization has been shown to be obtained in quasipolynomial time, making these systems more efficient and a subject for proof-complexity research [BGGP15, Das12].

The syntax of open deduction derivations is as follows:

$$\begin{array}{c} A \\ \Downarrow \\ C \end{array} ::= A \mid \begin{array}{c} A_1 \\ \Downarrow \\ C_1 \end{array} \wedge \begin{array}{c} A_2 \\ \Downarrow \\ C_2 \end{array} \mid \begin{array}{c} C_1 \\ \Uparrow \\ A_1 \end{array} \to \begin{array}{c} A_2 \\ \Downarrow \\ C_2 \end{array} \mid \begin{array}{c} A \\ \Downarrow \\ \dfrac{B}{B'}\,\mathtt{r} \\ \Downarrow \\ C \end{array}$$

The arrows give the direction of a derivation. Downward arrows correspond to the standard derivation where the top formula is the *premise* and the bottom formula the *conclusion*. Upward arrows reverse the roles: in the case of the implication, the left hand side is a derivation from A_1 to C_1. A derivation from A to C can be a formula i.e. $A = C$, a conjunction of two derivations giving a derivation from $A = A_1 \wedge A_2$ to $C = C_1 \wedge C_2$, an implication giving a derivation from $A = C_1 \to A_2$ to $C = A_1 \to C_2$, or the vertical composition of two derivations using a rule `r`.

But their most interesting feature in the context of this thesis is *atomicity*, which makes possible to replace rules by their *atomic* restriction, thanks to a linearized distributivity rule called the *medial* rule `m` and the *switch* rule `s`:

$$\frac{(A \wedge B) \vee (C \wedge D)}{(A \vee C) \wedge (B \vee D)}\,\mathbf{m} \qquad \frac{A \wedge (B \vee C)}{(A \wedge B) \vee C}\,\mathbf{s}$$

In particular, unlike in the sequent calculus [Brü03b], it becomes possible to replace contractions $\triangle$ and cocontractions $\triangledown$ by their atomic version:

$$\frac{A}{A \wedge A}\,\triangle \qquad \frac{A \vee A}{A}\,\triangledown \qquad \frac{A \to B}{(A \to B) \wedge (A \to B)}\,\triangle \rightsquigarrow \frac{\dfrac{A}{A \vee A}\,\triangledown \to \dfrac{B}{B \wedge B}\,\triangle}{(A \to B) \wedge (A \to B)}\,\mathtt{m}$$

The transformation above illustrates how the contraction of an implication $A \to B$ is replaced by contractions on smaller subformulas A and B. By repeating this process, we are eventually able to get a proof where inference rules are solely applied to atomic formulas.

The first Curry-Howard style interpretation of deep inference was obtained for a calculus with an intuitionistic natural deduction system, introduced in [BL05, BM08]. This calculus was introduced as a new way to describe derivations in Guglielmi's Formalism A, with reduction rules removing bureaucracy (i.e. identifies two morally equivalent proofs), in order to address cut-elimination. Terms (called *proof terms*) are essentially a notation for proofs, so instead of *variables, abstractions, applications*, they consider *identity*, `rule`, *composition, conjunction, implication*, where `rule` is one of the infer-

ence rules of intuitionistic logic. Unlike in the λ-calculus, function composition is used instead of application. Their calculus is closely related to categorical combinatorics, and thus can be seen as a rewriting theory on the syntax of cartesian closed categories. However, it does not satisfy preservation of strong normalization.

Originating from computational considerations, a typed λ-calculus with *explicit sharing*, the atomic λ-calculus [GHP13] was developed to correspond to an intuitionistic deep inference system via a Curry-Howard-style isomorphism. This calculus is one of the two pillars on which the thesis builds. The syntax is as follows:

$$\begin{aligned}
&\textit{Terms } t, u ::= \ x \mid \lambda x.t \mid (t)u \mid u[\phi] \\
&\textit{Closures } [\phi], [\psi] ::= \ [\vec{x_p} \leftarrow t] \mid [\vec{x_q} \twoheadleftarrow \lambda y.t^q] \\
&\textit{Tuples } t^p ::= \langle\, t_1, \ldots, t_p \,\rangle \mid t^p[\phi]
\end{aligned}$$

This calculus refines the λ-calculus with a *sharing* constructor $[x_1, \ldots, x_n \leftarrow t]$ corresponding to contraction (as in explicit substitution-calculi [ACCL91]), a *distributor* constructor $[x_1, \ldots, x_n \twoheadleftarrow \lambda y.t^n]$, a computational interpretation of the medial rule allowing us to perform *atomic reduction steps*, i.e. duplications of subterms independently of their context, and the use of unique variable names such that the β-reduction is implemented by a *linear* substitution. The β-reduction $(\lambda x.u)t \longrightarrow_\beta u\{t/x\}$ substituting t for each of the p occurrences of the variable x in u, becomes $(\lambda x.u[x_1, \ldots, x_p \leftarrow x])t \rightsquigarrow_\beta u[x_1, \ldots, x_p \leftarrow t]$ in the atomic calculus, where t is bound to the variables $x_1, \ldots, x_p$ representing the distinct occurrences of x. The duplication of t is then carried out atomically, one constructor at a time, by separate rules. In the following example, after the first reduction, we share $\lambda y.v$ with $x_1, \ldots, x_p$ in u, then we *freeze* λy (as shown by $\twoheadleftarrow$) while replicating v p-times in a tuple $\langle v_1, \ldots, v_p \rangle$, then we distribute λy over $\langle v_1, \ldots, v_p \rangle$ to obtain p copies of $\lambda y.v$, thus duplicating independently the body v of its constructor λy:

$$\begin{aligned}
(\lambda x.u[x_1, \ldots, x_p \leftarrow x])\ \lambda y.v &\rightsquigarrow_\beta u[x_1, \ldots, x_p \leftarrow \lambda y.v] \\
&\rightsquigarrow_* u[x_1, \ldots, x_p \twoheadleftarrow \lambda y. \\
&\qquad \langle v_1, \ldots, v_p \rangle [y_1, \ldots, y_p \leftarrow y]] \\
&\rightsquigarrow \ u\{\lambda y_1.v_1/x_1\}\ \ldots \\
&\qquad \{\lambda y_p.v_p/x_p\}
\end{aligned}$$

This calculus underlines the connection between atomicity and optimal graphs [Lam90], implements fully-lazy sharing [Bal12] (avoiding duplication of constant parts of an

expression), while preserving the principal properties of the λ-calculus.

1.2.5 The $\lambda\mu$-calculus and the classical Curry-Howard-Lambek correspondence

Extending the Curry-Howard-Lambek correspondence to classical logic leads to a radical reunderstanding of the notion of computation, as it was widely believed that classical logic had no computational meaning. This belief can be illustrated by the drinker's principle, due to Smullyan, a tautology that cannot be proved in intuitionistic logic. Informally, it states that in every (nonempty) bar, there is a customer (the drinker), such that if he drinks, then everybody drinks. To prove it, we can split the proof in two cases, either everybody drinks, or there is at least one person who doesn't drink. If everybody drinks, then picking anybody makes the formula true, so we are done. Now if at least one person is not drinking, he can be taken as a witness. Only classical logic can prove this, since an intuitionistic proof would have to be able to construct a witness, while classical logic can "change its mind". This power of classical logic can be added through different rules, through the law of excluded middle, Peirce's law, or in sequent calculus right contractions. Each illustrates a different aspect of classicism.

In the natural presentation of the intuitionistic Curry-Howard isomorphism, the proof systems allow only one conclusion. When we remove this restriction and allow multiple conclusions ($\Gamma \vdash \Delta$ instead of $\Gamma \vdash A$), one may have the possibility to follow different paths in the cut-elimination procedure, thus obtaining several distinct proofs without cuts of the same judgment. This issue is known as Lafont's critical pair [GTL89]:

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \vdots\\ \Gamma \vdash \Delta\end{array}}{\Gamma, A \vdash \Delta}\,\texttt{Lwk} \qquad \dfrac{\begin{array}{c}\Pi_2\\ \vdots\\ \Gamma' \vdash \Delta'\end{array}}{\Gamma' \vdash \Delta', A}\,\texttt{Rwk}}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,\texttt{Cut}$$

Here the cut-elimination can choose between the subproofs Π_1 and Π_2, thus this proof can reduce to two distinct normal forms, making cut-elimination non confluent. To conserve the nice properties of intuitionistic logic, these two proofs should be identified. However doing so leads to a collapse, and reduces the system to a boolean algebra.

Therefore, no interpretation of classical logic was known until Griffin showed an analogy between classical logic and Felleisen's control operator $\mathcal{C}$ [Gri90]. In the Curry-Howard correspondence's light, the law of excluded middle becomes an operator (a continuation) on the control flow of the program (such as Scheme's `call/cc`, or exceptions in

imperative languages). The two different proofs in Lafont's example become the proofs corresponding to two evaluation strategies, namely call-by-name and call-by-value, i.e.:

1. An abstraction is never reduced
2. In an application (`Abstraction`)`Argument`, reduce this redex first in call-by-name, reduce `Argument` first in call-by-value
3. Otherwise, reduce `Argument`.

In a categorical view, to get an interpretation of classical logic we need to abandon cartesian closedness, either by breaking the duality between product and coproduct (losing the cartesian product), or by dropping currying (losing closedness). Another possibility is to give up the symmetry between $\wedge$ and $\vee$ by having a cartesian product but not a real sum, and we will focus on this approach. Removing the duality between $\wedge$ and $\vee$ corresponds to choosing to reduce one of the two branches. The choice of a branch is consistent and enforced by the type system. Types govern the reduction strategy, and in that way enforce confluence. One way to make this choice is by introducing polarities for formulas, with reduction depending on the polarities, as in Girard's LC [Gir91]. In this system a sequent, to be valid, must have zero or one positive formula (in the *stoup*). This idea of selecting a distinguished formula in each sequent is also at the core of Parigot's $\lambda\mu$-calculus, the other main pillar of this thesis.

The whole correspondence has been established shortly after by Parigot [Par92], who developed a Curry-Howard interpretation of a classical natural deduction with multiple conclusions known as the *$\lambda\mu$-calculus*. It extends the λ-calculus while keeping the properties of confluence, preservation of strong normalization, and in a typed setting, subject reduction and strong normalization. The introduction of classical rules is done through the *μ-abstraction* constructor, that deals with new kinds of variables known as μ-variables (denoted by α, β). In the logical framework, μ-variables are indexing the formulas among the multiple conclusions.

There is a distinction between *unnamed* terms t, and *named* terms of the form $n = (t)\beta$, where t is unnamed, and unnamed terms are inductively defined by the following syntax:

$$t, u ::= \; x \mid \lambda x.t \mid (t)u \mid \mu\alpha.(t)\beta$$

A structural rule to reduce applications of μ-abstractions to terms is added to the calculus:

$$(\mu\alpha.n)t \to_\mu \mu\alpha.n\{(w)\, t\alpha/(w)\alpha\}$$

An intuitive way to interpret this rule is to see it as recursively passing the argument t, to each subterm w that has been named with the μ-variable α. From a computational perspective, the μ constructor abstracts over continuations, and can be seen as a potentially infinite λ-abstraction. Consider the following reduction:

$$((\mu\alpha.n)\, t_1) \dots t_p \to^*_\mu \mu\alpha.n\{(((w)\, t_1) \dots t_p)\alpha/(w)\alpha\}$$

The idea is to apply an arbitrary number p of arguments to the subterms w named with α. A μ-abstraction can thus be viewed as an arbitrary number of λ-abstractions. Similarly, in the application of an unnamed term t to a μ-variable α, the latter can be seen as an infinite stream of inputs.

A judgment $\Gamma \vdash A \mid \Delta$ consists of a list of formulas Γ annotated with λ-variables, a distinguished conclusion A typing an unnamed term, and a list Δ of formulas labeled with μ-variables. We denote by $\neg A$ the implication $A \to \bot$ (where $\bot$ denotes falsity). The type system for the $\lambda\mu$-calculus is given below:

$$\frac{}{x : A \vdash A}\ \text{Var} \qquad \frac{t : \Gamma, A \vdash B \mid \Delta}{\lambda x.t : \Gamma \vdash A \to B \mid \Delta}\ \lambda$$

$$\frac{t : \Gamma \vdash A \to B \mid \Delta \qquad u : \Gamma' \vdash A \mid \Delta'}{(t)u : \Gamma, \Gamma' \vdash B \mid \Delta, \Delta'}\ @$$

$$\frac{t : \Gamma \vdash A \mid \Delta}{(t)\alpha : \Gamma \vdash \bot \mid A^\alpha, \Delta}\ @_n \qquad \frac{(t)\beta : \Gamma \vdash \bot \mid A^\alpha, \Delta}{\mu\alpha.(t)\beta : \Gamma \vdash A \mid \Delta}\ \mu$$

Parigot's $\lambda\mu$-calculus follows a call-by-name strategy, and successfully links classical constructions to control operators. Felleisen's $\mathcal{C}$ operator is akin to *call/cc*, a control operator that captures the current continuation (i.e. the "frozen" programming context that remains to be executed), making it possible to resume execution later. These continuations can be seen as transformations of λ-terms following an evaluation strategy, after applying a *continuation passing style (CPS) translation* [Plo75].

Example 1.2.1. An example is computing `factorial(n)`, then use the result for another function (e.g. sum, inverse). The other function will be the continuation of the current program. Concretely, if $\phi\colon \mathbb{N} \to \mathbb{N}$ is the factorial function $\phi(n) = n!$, its CPS translation ϕ^* is the following function, with the following type, for any A.

$$\phi^*(n, f) = f(\phi(n)) \qquad \phi^*\colon \mathbb{N} \times (\mathbb{N} \to A) \to A$$

The function f is the *continuation*, and ϕ^* will use f as an argument to compute the result of $f(\phi(n))$.

Note that if ϕ has type $A \to B$, if f has type $B \to C$, then ϕ^* has type $A \to (B \to C) \to C$. Replacing C by $\bot$ transforms B to its double negation translation $\neg\neg B$. Hence a CPS translation given a certain evaluation strategy is akin a to double negation translation (i.e. an embedding of classical formulas into an intuitionistic setting). Passing continuations as arguments gives the possibility to exit a program, or go/jump to another part of the program. They are interesting for recursive calls: the continuation can be stored and resumed after the recursive call has returned a certain value.

Many other variants of the $\lambda\mu$-calculus were developed, with the aim of obtaining good computational properties. One of the most notable is Curien and Herbelin's $\bar{\lambda}\mu\tilde{\mu}$ [CH00]. For development in this dissertation, an important variant is Saurin's $\Lambda\mu$-calculus [Sau08]. Its aim is to satisfy Böhm's theorem, which fails in the $\lambda\mu$-calculus [DP01]. The syntax of the $\Lambda\mu$ is as follows:

$$t, u ::= \; x \mid \lambda x.t \mid (t)u \mid (t)\alpha \mid \mu\alpha.t$$

It extends the $\lambda\mu$-calculus by adding former "names" $(t)\alpha$ to the syntax, providing more applicative contexts to separate non-equivalent terms. Intuitively, α represents a place where subprograms can be stored thanks to the construct $(M)\alpha$. Binding with μ then retrieves what has been stored by the variable α. Remarking that the μ constructor abstracts over continuations, he observes that it can be seen as a potentially "infinite" stream of λ-abstractions. He then developed an extension including streams in the syntax [GS10], the $\Lambda\mu S$-calculus, developed in the next chapter.

Once the Curry-Howard correspondence had been established for classical logic, work started on the categorical aspect of control operators. From a categorical perspective, Lafont's critical pair corresponds to Joyal's theorem, which states that adding the requirement of a bifunctorial disjunction to a categorical model of intuitionistic logic (a CCC) is equivalent to the category being a boolean algebra. Building most notably on the works [Ong96, HS97], the solution was found by Selinger [Sel01], who found a categorical correspondence between the $\lambda\mu$-calculus and cartesian closed categories with a premonoidal structure. The categorical model in fact splits into two equivalences, the control categories, whose internal language corresponds to the call-by-name evaluation strategy, and the cocontrol categories whose internal language corresponds to the call-by-value evaluation strategy. This correspondence sheds a new light on the computational content of classical logic, since it also gave a syntactic duality, as there are mutually inverse translations between the two strategies, which preserve the operational semantics. While a control category has a bifunctorial product, its sum should not be interpreted as a bifunctor, but rather as a functor giving a premonoidal struc-

ture, consistently with the consequence of the choice of evaluation strategy. Conversely, in cocontrol categories the product is not a real product (i.e. not a bifunctor). Of note is that categorical semantics relies on a CPS translation, whose categorical account is a Yoneda embedding.

1.3 Motivation

In this section we address the main motivation for our work on the atomic $\lambda\mu$-calculus. The broad idea is to give greater control over duplication and deletion of terms, and thus efficiency of computation. It comes from explicit substitution calculi, which already enjoy a rich theory, the main concept being that of *sharing*. Informally, the idea behind sharing is that a resource used by multiple clients can be shared between them instead of being duplicated then given to each client. Sharing acts like a `let` operator/construct, similar to memoization for big step evaluation, and to call-by-need for small-step evaluation. For example, instead of computing

```
factorial(40) * factorial(40) + 8 * factorial(40)
```

which duplicates a heavy computation, we can share `factorial(40)` as follows:

```
let x = factorial(40) in x*x + 8*x
```

which makes it possible to avoid calculating `factorial(40)` several times, by making it a shared resource instead. The two programs are extensionally the same since they return the same result, but their operational semantics should be different. Various approaches exist to model formalisms with better control over duplication and deletion, such as λ-calculi with explicit substitutions [ACCL91]. A notable extension is Lengrand's λlxr [Len06], a linear calculus with explicit substitutions, weakening and contraction, satisfying preservation of strong normalization and having constructors for duplication and deletion of explicit substitutions. It corresponds to a fragment of proof nets for intuitionistic linear logic. A framework including Lengrand's calculus is the prismoid of resources [KR09], relating several calculi in a unified presentation, each choosing the sorts of resources (weakening, contraction, substitution) to control. Another calculus is the *linear substitution calculus* [Acc12], a different approach to explicit substitutions which is able to capture the workings of several abstract machines [ABM14], in particular modeling different evaluation strategies.

The atomic λ-calculus, which enjoys a natural form of sharing, has been studied with

the same motives. It is an *explicit sharing* calculus in the sense that while it has many similarities with explicit substitution calculi, the reduction rules are very different. In particular, lifting rules are inverted. It satisfies *fully-lazy sharing*, first described by Wadsworth, in which the "constant" parts (the *maximal free subexpressions*) of a program unaffected by the instantiation of the function are not duplicated (they are shared instead), thus providing an efficient model of computation. This corresponds to the maximal level of sharing that can be naturally maintained during reduction in a term calculus extended with only sharings.

A natural way to model and implement sharing is via graphs. Whereas a λ-term is syntactically a tree, common subterms may become shared straightforwardly by representing the term instead as a directed acyclic graph (DAG) [Wad71].

1.3.1 Sharing graphs

Sharing is strongly related to *optimality*, introduced by Lévy [Lév80], and Lamping's optimal graphs [Lam90] implement Lévy's optimal reduction. A reduction strategy is optimal when all the elements in the same family of redexes are reduced in one step, and when there are no unnecessary steps of work created. Viewing terms as graphs, Lamping's approach is to share partial terms, i.e. terms with a hole. His graphs introduce not only sharing but also *unsharing* nodes: the *fan-in* gathers the pointers to a context, whereas the *fan-out* (unsharing) collects the possible ways to fill a hole. This allows sharing of subexpressions of a term, and atomic duplications, similarly to the atomic λ-calculus. Lamping showed that it was impossible for any reduction strategy to completely avoid unnecessary duplications, however sharing is a way to manage those duplications (by representing them as a single entity). Optimal graphs relate to encodings of intuitionistic logic into linear logic. A prototype interpreter has been implemented by Asperti [AG98] to show the workability of optimal graphs.

1.3.2 Continuations

The $\lambda\mu$-calculus links classical operators to control operators, which give the possibility to manipulate the execution environment (e.g. by capturing partial terms). For instance, one can implement exceptions with continuations. In practice, programming languages often use side-effects (i.e. changing a certain state or interacting with the outside world in addition to returning a value) such as exceptions. From the $\lambda\mu$-calculus, an interesting direction would be an extension modeling more general effects. A mathematical approach to side-effects has been studied through algebraic effects [FS14].

1.4 Thesis outline

In this section, we first give the general overview of this thesis, what motivates our work and describe the issues that influenced our technical choices. We then detail the content of each chapter.

The main contribution of this thesis is the atomic $\lambda\mu$-calculus, which extends the Curry-Howard correspondence between deep inference formalisms and term calculi to a classical setting. The result is a direct extension of the $\lambda\mu$-calculus and of the atomic λ-calculus, giving an explicit sharing λ-calculus with control operators. Similarly to the atomic λ-calculus, we can give a deep inference typing system, corresponding to term duplication on individual constructors, in the open deduction formalism. To obtain the syntax and rules, we applied a technique similar to that of the atomic λ-calculus, giving first a $\lambda\mu$-calculus with explicit sharings, then a distributor for the μ-abstraction. One first difficulty is linked to the syntax and to the main reduction rule of the $\lambda\mu$-calculus. The $\lambda\mu$-calculus distinguishes two different kinds of terms, which complicates the syntax with sharings. Furthermore, a β-step can be easily expressed in terms of sharing or explicit substitutions $[x := t]$, whereas expressing a μ-reduction with $[(w)\alpha := (w)t\alpha]$ would be problematic, since we want to modify all subterms of the form $(w)\alpha$. For these reasons we work on a variant, the $\Lambda\mu S$-calculus [GS10], with one kind of term and considering a new *stream* application compatible with sharings. Another challenge appears when defining the type system in open deduction. The deep inference methodology allows us to identify object (connectives) and meta (commas in sequents, branching derivations) levels, and classical logic requires working with multiple conclusions and distinguishing one main conclusion, thus introducing another meta-level (corresponding to a meta disjunction $\vee$). Therefore we need to find a suitable formulation to work with classical natural deduction, allowing a clear computational reading, while keeping everything at the same level, in the simplest way (introducing as few new rules as possible, keeping proofs as short as possible). We show that our resulting calculus, the atomic $\lambda\mu$-calculus, satisfies confluence, subject reduction in a typed setting, and preservation of strong normalization (PSN). The proof for PSN, the main theorem for our calculus, is divided into two parts, with a intermediate *weakening* calculus that helps isolate the hard part of the proof. This part requires finding a reduction strategy which provides an infinite path for a term whenever there exists one, while keeping infinite reductions outside of weakenings, thus interpreted to infinite paths in the $\Lambda\mu S$-calculus. Our approach to this strategy is different from that in [GHP13], relying on the property the weakening calculus preserves any β, μ infinite path. Another necessary lemma to show PSN is the strong normalization of sharing

reductions. We present a new proof for this lemma, following an approach that matches more closely the idea of sharing. As for the atomic λ-calculus, we have full laziness.

In Chapter 2 we define our atomic $\lambda\mu$-calculus $\Lambda\mu S^a$, giving a correspondence with classical open deduction. Choosing to extend $\Lambda\mu S$ instead of other $\lambda\mu$-calculi originates from several difficulties, related to the syntax and the typing system, that we describe in this chapter. We extend the syntax of the atomic λ-calculus and the $\Lambda\mu S$-calculus with sharings (respectively distributors) of streams S (respectively μ-abstractions $\mu\alpha.t$), and a set of rules symmetric to the intuitionistic case appearing with the μ-reduction. We then describe the corresponding typing system in natural deduction (in sequent-style) and the same rules in open deduction.

Chapter 3 gives several properties of the atomic $\lambda\mu$-calculus, showing that reductions and translations are well-behaved with respect to the $\Lambda\mu S$-calculus. These properties will then be used to prove PSN.

Chapter 4 introduces an auxiliary calculus, the weakening calculus, a bridge between the $\Lambda\mu S$-calculus and the atomic calculus that helps prove PSN for $\Lambda\mu S_a$ with respect to the $\Lambda\mu S$-calculus. A difficulty in the proof of PSN appears with reductions happening in nullary sharings (or weakenings), thus simulating a reduction step in $\Lambda\mu S^a$ with no reduction step in $\Lambda\mu S$. This leads to constructing the weakening calculus. From there, we show PSN for the weakening calculus with respect to the $\Lambda\mu S$-calculus by using a perpetual or exhaustive strategy, which guarantees to find a infinite path in the $\Lambda\mu S$-calculus from an infinite path in the weakening calculus. This strategy differs from the perpetual strategy in the atomic λ-calculus, adapting to the properties of the weakening calculus.

Chapter 5 shows that sharing reductions are strongly normalizing. The idea for this proof is new, and follows a graphical intuition behind terms. After a sharing reduction, the number of copies of subterms can decrease, so can the lengths of paths from the root to closures. We thus build a strictly decreasing measure based on these parameters.

Chapter 6 states and shows PSN for the $\Lambda\mu S_a$-calculus with respect to the $\Lambda\mu S$-calculus, i.e. the atomic translation of a strongly normalizing $\Lambda\mu S$-term is strongly normalizing.

Chapter 7 focuses on the full-laziness of our calculus. We prove that it is possible to share the maximal constants parts of a term, eventually evaluating them only once, therefore limiting duplications to the remaining parts of the term.

Chapter 2

The atomic lambda-mu-calculus

In this chapter we present the steps to build the atomic $\lambda\mu$-calculus. Our aim is to obtain a calculus with explicit sharing, enjoying the same properties as the atomic λ-calculus and the $\lambda\mu$-calculus: confluence, full laziness, preservation of strong normalization, and in a typed setting, subject reduction.

We will first try to naively extend Parigot's $\lambda\mu$-calculus to an atomic setting, and see that this does not work well with μ-reduction. Adding explicit sharings to $\lambda\mu$-terms is straightforward: multiple occurrences of variables are given fresh names and become bound to a common variable inside a sharing. For example, $\mu\beta.(\mu\alpha.(x)\beta)\beta$ becomes $\mu\beta.(\mu\alpha.(x)\beta_1)\beta_2[\beta_1, \beta_2 \leftarrow \beta]$. In an atomic setting, while β-reduction translation is straightforward, μ-reduction cannot easily be expressed with sharings. The β-reduction $(\lambda x.t)u \longrightarrow t\{u/x\}$ is interpreted as $(\lambda x.t[x_1, \ldots, x_n \leftarrow x])u$ and reduces to $t[x_1, \ldots, x_n \leftarrow u]$. For μ-reduction, take the following example:

$$(\mu\alpha.(\mu\beta.(x)\alpha)\alpha)t \rightarrow_\mu \mu\alpha.(\mu\beta.(x)\ t\ \alpha)t\ \alpha$$

$$(\mu\alpha.(\mu\beta.(x)\alpha_1)\alpha_2[\alpha_1, \alpha_2 \leftarrow \alpha])t \rightarrow_\mu ???$$

The μ-reduction rule $(\mu\gamma.t)u \longrightarrow_\mu \mu\gamma.t\{(w)u\gamma/(w)\gamma\}$ modifies the structure of the terms of the form $(w)\gamma$ by pattern-matching on the application to γ, which makes it difficult to express it in terms of explicit substitutions or sharings. In general, we would need something of the form $t[(w_1)\gamma_1, \ldots, (w_n)\gamma_n \leftarrow (w)u\gamma]$, shifting away from the structure of sharings or explicit substitutions and making the syntax more complicated.

We need a slightly different approach, where our syntax should be as close as possible to that of the atomic λ-calculus. Intuitively, we would like the previous reduction to

look like:

$$(\mu\alpha.(\mu\beta.(x)\alpha_1)\alpha_2[\alpha_1, \alpha_2 \leftarrow \alpha])t \to_\mu (\mu\alpha.(\mu\beta.(x)\alpha_1)\alpha_2[\alpha_1, \alpha_2 \leftarrow (t)\alpha]) \qquad (\star)$$

but the right term is incorrect, since application is left-associative.

This can be fixed by introducing a new, right-associative, *stream* application $\circ$. Such a variant has been studied by Saurin and Gaboardi [GS10]. In this calculus, there are two applications, the usual one, and the stream application. A stream is similar to a list, and the μ-reduction rule becomes $(\mu\alpha.t)u \longrightarrow_\mu \mu\beta.t\{u \circ \beta/\alpha\}$, where instead of successively applying terms annotated by α to u, we add u to a stream of terms.

In this case, the previous example becomes:

$$(\mu\alpha.(\mu\beta.(x)\alpha)\alpha)t \to_\mu \mu\alpha.(\mu\beta.(x)\ (t \circ \alpha))(t \circ \alpha)$$

$$(\mu\alpha.(\mu\beta.(x)\alpha_1)\alpha_2[\alpha_1, \alpha_2 \leftarrow \alpha])t \to_\mu (\mu\alpha.(\mu\beta.(x)\alpha_1)\alpha_2[\alpha_1, \alpha_2 \leftarrow t \circ \alpha])$$

In the general case, $(\mu\alpha.t[\alpha_1, \ldots, \alpha_n \leftarrow \alpha])u$ reduces to $\mu\alpha.t[\alpha_1, \ldots, \alpha_n \leftarrow u \circ \alpha]$.

We thus need to extend our syntax with new *stream* terms:

$$S, T ::= \alpha \mid t \circ S$$

and μ-abstractions of the form $\mu\alpha.(t)S$.

The $\lambda\mu$-calculus already distinguishes terms from names, and extending it with sharings adds streams, now giving three different expressions, thus burdening the study of the calculus. Therefore we can consider Saurin's variant, the $\Lambda\mu$-calculus, which includes names in the term syntax, then extend it with streams to get the $\Lambda\mu S$-calculus:

$$t, u ::= x \mid \lambda x.t \mid (t)u \mid (t)S \mid \mu\alpha.t$$

The syntax of the $\Lambda\mu S$-calculus with sharings can then be naturally extended to an atomic setting, giving the $\Lambda\mu S_a$-calculus.

To deal with μ-reductions, we can now introduce a μ-distributor analogous to the λ-distributor for λ-abstractions. Reduction rules involving μ are then close to those involving β. In particular, in a μ-abstraction, the distributor works similarly and allows us to duplicate the μ-constructor independently from the body of the abstraction.

The next step is to construct a suitable type system in open deduction. Working in sequent-style, we can define a type system for $\Lambda\mu S_a$-terms with multiple conclusions

(as in Parigot's classical natural deduction). However, linking atomic terms to their sequent-style proof is not very intuitive, especially if we look at sharing and distributor rules. Those two rules can be naturally described in open deduction, which corresponds via Curry-Howard to the atomic λ-calculus, and therefore is the best choice to represent our typed terms.

We can then attempt to build a multi-conclusion open deduction system. In classical natural deduction, sequents are of the form:

$$\Gamma \vdash A \mid \Delta$$

with at most one *main conclusion*, distinguished from the others by a $\mid$ meta-connective, corresponding to a special disjunction.

In open deduction, we also need a way to specify the main conclusion on which to apply an inference rule. We thus first introduce the $\mid$ connective in open deduction, then add new *switch* rules:

$$\frac{(A \mid \Delta) \wedge (B \mid \Delta')}{(A \wedge B) \mid \Delta \vee \Delta'}\,\mathsf{s}_1 \qquad \frac{A \to (B \mid \Delta)}{(A \to B) \mid \Delta}\,\mathsf{s}_2$$

to retrieve our main conclusion.

There are several reasons why this system is not convenient. First, in proofs for atomic λ-terms, premises are conjunctive formulas of the form $\Gamma \wedge \Gamma'$ (indexing λ-variables), and conclusions are formulas of the form $A \to B$ (indexing a term). Ideally our proofs should follow a similar structure, but multiple conclusions with disjunctions prevent this. Another problem is that rules λ, @ and $@_n$ require the use of switch rules before being applied to the main conclusions, which leads to very long proofs. Last, and this is the main reason, reducing bureaucracy is the very essence of deep inference and open deduction, and multiple conclusions forces to introduce another meta-level with the $\mid$ connective.

Consider for instance the classical formula $\neg\neg A \to A$. In a system with multiple

conclusions, the proof would look like this:

$$\dfrac{\dfrac{(\neg A \mid A^\alpha) \wedge (\top \mid \bot^\phi)}{\dfrac{(\neg A \mid A^\alpha)}{\dfrac{A^x \wedge (\neg A \mid A^\alpha)}{\dfrac{A^x \wedge \neg A}{\neg\neg A^y \wedge A^x \to \dfrac{\bot}{\dfrac{\bot \mid A^\alpha, \bot^\delta}{\dfrac{\bot \mid A^\alpha}{\dfrac{(A^x \to \bot) \mid A^\alpha}{\neg\neg A^y \to \dfrac{\dfrac{\neg\neg A^y \wedge (\neg A \mid A^\alpha)}{\dfrac{\neg\neg A^y \wedge \neg A}{\dfrac{\bot}{\dfrac{(\bot \mid A^\alpha) \wedge (\top \mid \bot^\phi)}{\dfrac{\bot \mid A^\alpha, \bot^\phi}{A \mid \bot^\phi}\mu}\mathsf{s}_1}\mathsf{w}}\mathsf{s}_1}\mathsf{s}_1}\mathsf{s}_2}\mu}\mathsf{w}}\mathsf{s}_1}\lambda}\lambda}\wedge (\top \mid \bot^\phi)}{\dfrac{\neg\neg A^y \to (A \mid \bot^\phi)}{(\neg\neg A^y \to A) \mid \bot^\phi}\mathsf{s}_2}$$

Multiple conclusions (or disjunctions) appear with free μ-variables, and a non-main conclusion C^γ becomes main when γ is bound by the abstraction $\mu\gamma$.

To simplify the type system, a possibility is not to display those non-main conclusions until they get bound by a μ-abstraction. This way our system becomes more economical by avoiding the superfluous switch rules. The proof of $\neg\neg A \to A$ thus becomes:

$$\dfrac{\dfrac{\top^\phi}{\dfrac{\neg\neg A^y}{\dfrac{\neg A}{\dfrac{A^x \wedge \neg A}{\neg\neg A^y \wedge A^x \to \dfrac{\dfrac{A^x \wedge \neg A}{\dfrac{\bot}{\bot^\delta}\mathsf{w}}\mathsf{w}}{\neg\neg A^y \to \dfrac{\dfrac{\neg\neg A^y \wedge \neg A}{\dfrac{\bot}{\dfrac{\bot \wedge \top^\phi}{\bot}\mathsf{w}}\mathsf{w}}\mathsf{w}}{A^\alpha}\mathsf{w}}\mathsf{w}}\mu}\lambda}\lambda}\wedge \top^\phi \quad \vee A^\alpha}{\neg\neg A^y \to A}$$

As a consequence, μ-abstraction becomes very similar to λ-abstraction:

$$\lambda x.t \equiv \dfrac{\dfrac{\Gamma}{\Gamma \wedge A^x}\,\lambda}{A^x \rightarrow \quad \begin{array}{c} \mathsf{t} \\ \Downarrow \\ C \end{array}} \qquad \mu\alpha.t \equiv \dfrac{\dfrac{\dfrac{\Gamma}{\Gamma \wedge \neg A}\,\mu}{\begin{array}{c} \mathsf{t} \\ \Downarrow \\ \bot \end{array} \quad \vee A^\alpha}}{\;\;A\;\;}$$

In a λ-abstraction the part $A^x \rightarrow$ remains untouched during the derivation, and likewise, in a μ-abstraction the part $\vee A^\alpha$ does not interact with the rest of the proof, thus we can ignore this part until there is a μ-abstraction on α.

This chapter starts by presenting the basic $\Lambda\mu S_a^-$-calculus with sharings, which leads to the atomic $\Lambda\mu S_a$-calculus. We detail the syntax of $\Lambda\mu S_a$, its reduction rules, and typing system.

2.1 The basic $\Lambda\mu S$-calculus: $\Lambda\mu S_a^-$

We now work with an extension of Saurin's $\Lambda\mu$-calculus, the $\Lambda\mu S$-calculus with streams. The syntax of $\Lambda\mu S$ is defined as follows:

Definition 2.1.1. [Syntax of $\Lambda\mu S$]

$$\begin{aligned} &\textit{Terms } T, U ::= \; x \mid \lambda x.T \mid (T)U \mid (T)\mathcal{S} \mid \mu\alpha.T \\ &\textit{Streams } \mathcal{S} ::= \; \alpha \mid T \circ \mathcal{S} \end{aligned}$$

The β and μ-reduction rules can be applied to terms or streams, and therefore we obtain four rules:

Definition 2.1.2. [Reduction rules for $\Lambda\mu S$]

1. $(\lambda x.T)U \longrightarrow_{\beta_t} T\{U/x\}$
2. $(\lambda x.T)(U \circ \mathcal{S}) \longrightarrow_{\beta_s} (T\{U/x\})\mathcal{S}$
3. $(\mu\beta.T)U \longrightarrow_{\mu_t} \mu\beta.T\{(U \circ \beta)/\beta\}$
4. $(\mu\beta.T)\mathcal{S} \longrightarrow_{\mu_s} T\{\mathcal{S}/\beta\}$

Remark 2.1.3. We often use $\longrightarrow_{\beta,\mu}$ to say that we refer to one of these rules.

We now want to extend the $\Lambda\mu S$-calculus with explicit sharing in a way that captures the reduction rules above. The idea of sharing is to postpone the duplication of a term as much as possible by using a single shared representation of this term. All the copies of this term can then be evaluated simultaneously before being duplicated. In this section we present the syntax of the basic calculus $\Lambda\mu S_a^-$ with explicit sharings, and how to translate $\Lambda\mu S$ terms into $\Lambda\mu S_a^-$.

2.1.1 Syntax of $\Lambda\mu S_a^-$

We now define the $\Lambda\mu S_a^-$-calculus, extending the $\Lambda\mu S$ calculus with sharings $[\vec{x_p} \leftarrow t]$ for terms (respectively $[\vec{\gamma_p} \leftarrow S]$ for streams), where $\vec{x_p}$ denotes $x_1, \ldots, x_p$ (respectively $\vec{\gamma_p}$ denotes $\gamma_1, \ldots, \gamma_p$). The syntax of $\Lambda\mu S_a^-$ is defined as follows:

Definition 2.1.4. [Syntax of $\Lambda\mu S_a^-$]

$$\begin{aligned}
&\textit{Terms } t, u ::= \; x \mid \lambda x.t \mid (t)u \mid (t)S \mid \mu\alpha.t \mid u[\phi] \\
&\textit{Streams } S, T ::= \; \alpha \mid t \circ S \mid S[\phi] \\
&\textit{Closures } [\phi], [\psi] ::= \; [\vec{x_p} \leftarrow t] \mid [\vec{\gamma_p} \leftarrow S]
\end{aligned}$$

The following conditions must hold:

- in $[\vec{x_p} \leftarrow t]$ and $[\vec{\gamma_p} \leftarrow S]$ the variables x_p, γ_p are *binding*,
- in $\lambda x.t$ (respectively $\mu\alpha.t$) the variable x (respectively α) binds in t, and in $u[\phi]$ (respectively $T[\phi]$), the binding variables of $[\phi]$ bind in u (respectively T),
- a variable occurs exactly once.

To reduce the number of cases in definitions and proofs, we will introduce new notations. These notations are for $\Lambda\mu S_a^-$-terms t, but we use the same conventions for $\Lambda\mu S$-terms T.

Notation 1.
- χ denotes λ or μ-variables,
- u^* denotes terms and streams,
- $@(t, u^*)$ denotes applications $(t)u^*$ and $t \circ S$,
- $\mathcal{A}x.t$ denotes abstractions $\lambda x.t$ and $\mu\alpha.t\{\alpha/x\}$,
- $[\Phi]$ denotes a sequence of closures $[\phi_1] \ldots [\phi_n]$.

These notations will allow us to treat terms and streams uniformly where required, as well as our four applications and our two abstractions.

2.1.2 Translation $\Lambda\mu S \xrightarrow{(\!|-|\!)} \Lambda\mu S_a^-$

We translate the $\Lambda\mu S$-terms into linear terms in $\Lambda\mu S_a^-$. All occurrences $x_1, \ldots, x_p$ (respectively $\alpha_1, \ldots, \alpha_p$) of a same variable x (respectively α) are collected with a sharing $[x_1, \ldots, x_p \leftarrow x]$ (respectively $[\alpha_1, \ldots, \alpha_p \leftarrow \alpha]$). A term $t\frac{p}{x}$ (respectively $t\frac{p}{\alpha}$) corresponds to the term t where the p occurrences of x (respectively α) have been replaced with $x_1, \ldots, x_p$ (respectively $\alpha_1, \ldots, \alpha_p$). We first define the auxiliary translation $(\!|-|\!)'$, which collects occurrences of a bound variable x (respectively α) with a sharing $[x_1, \ldots, x_n \leftarrow x]$ (respectively $[\alpha_1, \ldots, \alpha_n \leftarrow \alpha]$):

Definition 2.1.5. [Translation $(\!|-|\!)'$]

- $(\!| x |\!)' = x$
- $(\!| \alpha |\!)' = \alpha$
- $(\!| @(T, U^*) |\!)' = @((\!| T |\!)', (\!| U^* |\!)')$
- $(\!| \mathcal{A}x.T |\!)' = \begin{cases} \mathcal{A}x.(\!| T |\!)' & \text{if } |T|_x = 1 \\ \mathcal{A}x.((\!| T\frac{p}{x} |\!)'[\vec{x_p} \leftarrow x]) & \text{if } |T|_x = p \neq 1 \end{cases}$

We can now define the translation $(\!|-|\!)$ from $\Lambda\mu S$-terms to $\Lambda\mu S_a^-$-terms, which also linearizes free variables:

Definition 2.1.6. [Translation $\Lambda\mu S \xrightarrow{(\!|-|\!)} \Lambda\mu S_a^-$] Let $x_1, \ldots, x_p, \alpha_1, \ldots, \alpha_k$ be the distinct variables occurring in t, such that $|T|_{x_i}, |T|_{\alpha_i} > 1$. Then:

$$\begin{aligned}(\!| T |\!) = &(\!| T\frac{l_1}{x_1} \ldots \frac{l_p}{x_p}\frac{l_{p+1}}{\alpha_1} \ldots \frac{l_{p+k}}{\alpha_k} |\!)'[(\vec{x_1})_{l_1} \leftarrow x_1] \ldots [(\vec{x_p})_{l_p} \leftarrow x_p] \\ &[(\vec{\alpha_1})_{l_{p+1}} \leftarrow \alpha_1] \ldots [(\vec{\alpha_k})_{l_{p+k}} \leftarrow \alpha_k]\end{aligned}$$

2.2 The atomic $\Lambda\mu$-calculus: $\Lambda\mu S_a$

We now extend our basic calculus with the distributor construct. The idea is that during reductions, we want to be able to duplicate smaller portions of a term instead

of copying the whole term. In particular, when duplicating abstractions, we would like to separate the body of the abstraction and duplicate it independently from its constructor. To do that we use the distributor construct $[\ldots \twoheadleftarrow \cdot]$, which contains a tuple of terms corresponding to the copies of the body of the abstraction to duplicate.

From a term $u[x_1, \ldots, x_p \leftarrow \mu\alpha.v]$, we want to eventually replace each x_i with $\mu\alpha.v$, and to do that we first create p copies $v_1, \ldots, v_p$ of v into a tuple $\langle v_1, \ldots, v_p \rangle$ while freezing the constructor $\mu\alpha$. Then we perform the substitution by distributing $\mu\alpha$ over the copies of v to obtain p copies of $\mu\alpha.v$.

$$\begin{aligned}(\lambda x.u[x_1, \ldots, x_p \leftarrow x])\ \mu\alpha.v &\rightsquigarrow_\beta u[x_1, \ldots, x_p \leftarrow \mu\alpha.v] \\ &\rightsquigarrow_* u[x_1, \ldots, x_p \twoheadleftarrow \mu\alpha.\langle v_1, \ldots, v_p \rangle[\alpha_1, \ldots, \alpha_p \leftarrow \alpha]] \\ &\rightsquigarrow \ u\{(\mu\alpha_1.v_1)/x_1\}\ \ldots\ \{(\mu\alpha_p.v_p)/x_p\}\end{aligned}$$

2.2.1 Syntax of $\Lambda\mu S_a$

The *atomic $\lambda\mu$-calculus* $\Lambda\mu S_a$ extends the basic calculus $\Lambda\mu S_a^-$ as follows:

Definition 2.2.1. [Syntax of $\Lambda\mu S_a$]

$$\begin{aligned}&\textit{Closures } [\phi], [\psi] ::= \cdots \mid [\vec{x_q} \twoheadleftarrow \lambda y.t^q] \mid [\vec{x_q} \twoheadleftarrow \mu\beta.t^q] \\ &\lambda\textit{-tuples } t^p ::= \langle\, t_1, \ldots, t_p \,\rangle \mid t^p[\phi]\end{aligned}$$

The conditions of $\Lambda\mu S_a^-$ apply, plus the following:

- In $\lambda x.t^p$ (respectively $\mu\alpha.t^p$) the variable x (respectively α) binds in t^p.

We denote by $\mathbb{T}$ the set of terms, by $\mathbb{S}$ the set of streams, and by $\mathbb{T}_p$ the set of p-terms. An expression $u^* \in \Lambda\mu S_a$ can be a term in $\mathbb{T}$, a stream in $\mathbb{S}$, or a p-term in $\mathbb{T}_p$. A variable χ can be a λ-variable in $\mathbb{T}$, or a μ-variable in $\mathbb{S}$. Let $\tau \in \mathbb{T} \cup \mathbb{S}$, a 0-ary sharing $[\leftarrow \tau]$ (resp. a 0-ary distributor $[\twoheadleftarrow \mathcal{A}y.t^0]$) is called a *weakening*.

Definition 2.2.2. Let $u^* \in \Lambda\mu S_a$. We define by induction on u^* the set $\mathcal{E}(u^*)$ of subexpressions of u^*:

1. $\mathcal{E}(x) = \{x\}$
2. $\mathcal{E}(\mathcal{A}x.t) = \{\mathcal{A}x.t\} \cup \mathcal{E}(t)$
3. $\mathcal{E}(@(t, t'^*)) = \{@(t, t'^*)\} \cup \mathcal{E}(t) \cup \mathcal{E}(t'^*)$

4. $\mathcal{E}(\langle\, t_1, \ldots, t_p \,\rangle) = \mathcal{E}(t_1) \cup \cdots \cup \mathcal{E}(t_p)$
5. $\mathcal{E}(u^*[\phi]) = \{u^*[\phi]\} \cup \mathcal{E}(u^*) \cup \mathcal{E}(\phi)$
6. $\mathcal{E}(\vec{\chi_q} \leftarrow \chi) = \mathcal{E}(\chi)$
7. $\mathcal{E}(\vec{x_q} \twoheadleftarrow \mathcal{A}x.t^p) = \mathcal{E}(t^p)$

2.2.2 Denotation $\Lambda\mu S_a \xrightarrow{[\![-]\!]} \Lambda\mu S$

We now define the translation from $\Lambda\mu S_a$ back to $\Lambda\mu S$. Terms are translated into terms by the *denotation* (or *interpretation*) function $[\![-]\!]$, whereas closures become substitutions which are translated by the auxiliary function $\{\!|-|\!\}$, which we write in place of a closure's usual square brackets [ ]. We write $\{\ldots/x_i\}_{i\leq p}$ whenever we apply multiple substitutions over each x_i. Let τ be a term or a stream in $\mathbb{T} \cup \mathbb{S}$, and χ_i be variables in $\mathbb{T} \cup \mathbb{S}$. The following translation (or denotation) maps $\Lambda\mu S_a$ to $\Lambda\mu S$:

Definition 2.2.3. [Denotation $\Lambda\mu S_a \xrightarrow{[\![-]\!]} \Lambda\mu S$]

- $[\![\, x \,]\!] = x$
- $[\![\, \alpha \,]\!] = \alpha$
- $[\![\, \mathcal{A}x.t \,]\!] = \mathcal{A}x.[\![\, t \,]\!]$
- $[\![\, @(t, \tau) \,]\!] = @([\![\, t \,]\!], [\![\, \tau \,]\!])$
- $[\![\, u^*[\phi] \,]\!] = [\![\, u^* \,]\!]\{\!|\, \phi \,|\!\}$
- $\{\!|\, \vec{\chi_p} \leftarrow \tau \,|\!\} = \{[\![\, \tau \,]\!]/\chi_i\}_{i\leq p}$
- $\{\!|\, \vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, t_1, \ldots, t_p \,\rangle[\Phi] \,|\!\} = \{(\mathcal{A}y.[\![\, t_i \,]\!]\{\!|\, \Phi \,|\!\})/x_i\}_{i\leq p}$
- $\{\!|\, \Phi \,|\!\} = \{\!|\, \phi_1 \,|\!\} \ldots \{\!|\, \phi_n \,|\!\}$ where $[\Phi] = [\phi_1] \ldots [\phi_n]$

2.2.3 Reduction rules

We can represent atomic terms as follows:

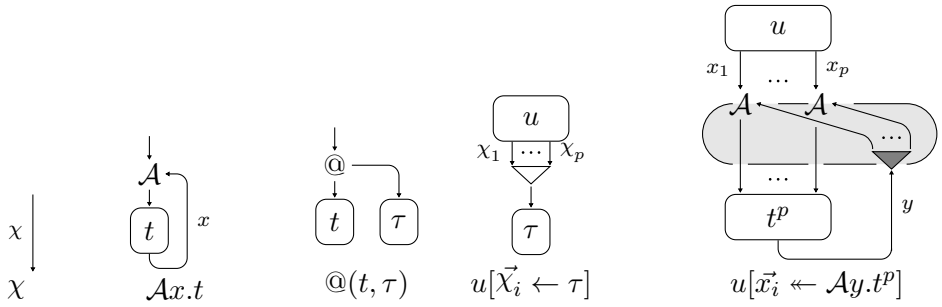
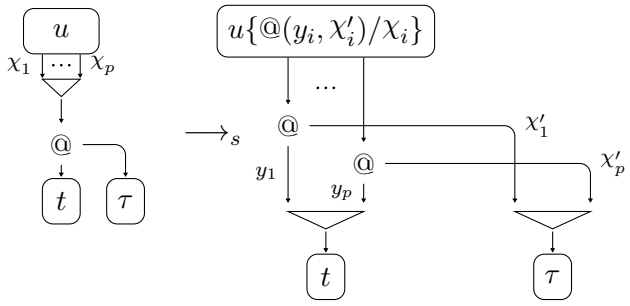
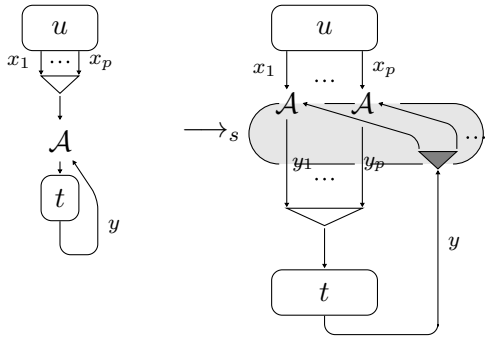


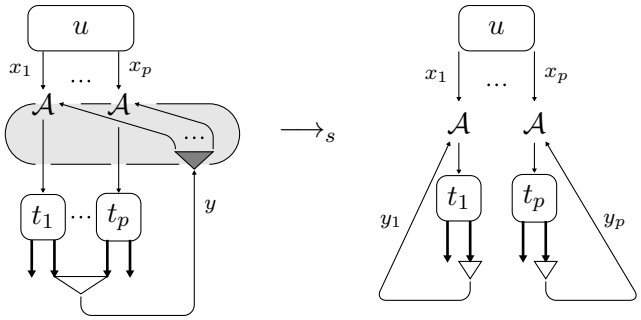
Figure 2-1: Graphical representation of atomic terms

The main innovation here comes from the atomic duplication rules: they allow us to copy smaller parts of a term instead of duplicating the whole term. Graphically, instead of duplicating an application $@(t, \tau)$, we duplicate t and τ separately:



The distributor rule enables the duplication of an abstraction by separating the abstraction constructor from the body. Graphically, instead of duplicating $\mathcal{A}x.t$, we create a tuple $\langle\, y_1, \ldots, y_p \,\rangle[\vec{y_i} \leftarrow t]$, then duplicate t separately (obtaining a tuple $\langle\, t_1, \ldots, t_p \,\rangle$), and finally distribute the abstraction constructors over the t_i:





where the thicker arrows represent a bundle of arrows.

The lifting rules push closures outside of the scope of a term, thus allowing us to postpone the duplication of a term, as in lazy reduction strategies. As a result (shown in Chapter 7), we get full laziness, i.e. a lazy strategy that can also duplicate parts of a term instead of copying the whole term.

Example 2.2.4. Let $T = \lambda a.(a)a$ and $U = \mu\gamma.((\lambda x.(x)x)y)\gamma$.

A possible step for $(T)U$ is to reduce to $(U)U$, duplicating U twice. Then we would need to eliminate redexes of U twice, which is something we want to avoid.

In the basic calculus,

$$t = (\!| T |\!) = \lambda a.(a_1)a_2[a_1, a_2 \leftarrow a]$$

and

$$u = (\!| U |\!) = \mu\gamma.((\lambda x.(x_1)x_2)y[x_1, x_2 \leftarrow x])\gamma$$

The reduction becomes:

$$(\lambda a.(a_1)a_2[a_1, a_2 \leftarrow a])\ u$$

$$\wr_\beta$$

$$(a_1)a_2[a_1, a_2 \leftarrow u]$$

where u is not duplicated but shared.

A possible step is then to β-reduce u:

$$(a_1)a_2[a_1, a_2 \leftarrow \mu\gamma.(\underbrace{(\lambda x.(x_1)x_2[x_1, x_2 \leftarrow x])y}\,)\gamma]$$

$$\wr_\beta$$

$$(a_1)a_2[a_1, a_2 \leftarrow \mu\gamma.((x_1)x_2[x_1, x_2 \leftarrow y])\gamma]$$

Then we can lift the sharing $[x_1, x_2 \leftarrow y]$ outside:

$$(a_1)a_2[a_1, a_2 \leftarrow \mu\gamma.((x_1)x_2[x_1, x_2 \leftarrow y])\gamma]$$

$$\wr_s^*$$

$$(a_1)a_2[a_1, a_2 \leftarrow \mu\gamma.((x_1)x_2)\gamma][x_1, x_2 \leftarrow y]$$

The main step is now to duplicate $\mu\gamma.((x_1)x_2)\gamma$.

$$(a_1)a_2[a_1, a_2 \leftarrow \mu\gamma.((x_1)x_2)\gamma]$$

$$\wr_s$$

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, v_1, v_2 \,\rangle[v_1, v_2 \leftarrow ((x_1)x_2)\gamma]]$$

This step *freezes* the μ-distributor, which is now independent from the rest of the term, and creates a tuple $\langle\, v_1, v_2 \,\rangle$, each v_i being shared with the term $((x_1)x_2)\gamma$ to duplicate.

Instead of duplicating the application $((x_1)x_2)\gamma$, we can break it into duplications of smaller subexpressions x_1, x_2 and γ.

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, v_1, v_2 \,\rangle[v_1, v_2 \leftarrow ((x_1)x_2)\gamma]]$$

$$\wr_s$$

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, (v_1)\gamma_1, (v_2)\gamma_2 \,\rangle[v_1, v_2 \leftarrow (x_1)x_2][\gamma_1, \gamma_2 \leftarrow \gamma]]$$

$$\wr_s$$

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, (v_1)w_1\gamma_1, (v_2)w_2\gamma_2 \,\rangle[v_1, v_2 \leftarrow x_1][w_1, w_2 \leftarrow x_2][\gamma_1, \gamma_2 \leftarrow \gamma]]$$

Since x_1 and x_2 are not bound, we can push sharings $[v_1, v_2 \leftarrow x_1]$ and $[w_1, w_2 \leftarrow x_2]$ outside of the scope of the distributor, to postpone duplications of x_1 and x_2 until necessary. To do that, we first apply an exchange rule, then push sharings outside:

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, (v_1)w_1\gamma_1, (v_2)w_2\gamma_2 \,\rangle[v_1, v_2 \leftarrow x_1]\underbrace{[w_1, w_2 \leftarrow x_2][\gamma_1, \gamma_2 \leftarrow \gamma]}]$$

$$\wr_s$$

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, (v_1)w_1\gamma_1, (v_2)w_2\gamma_2 \,\rangle[v_1, v_2 \leftarrow x_1][\gamma_1, \gamma_2 \leftarrow \gamma][w_1, w_2 \leftarrow x_2]]$$

$$\wr_s^*$$

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, (v_1)w_1\gamma_1, (v_2)w_2\gamma_2\,\rangle[\gamma_1, \gamma_2 \leftarrow \gamma]][v_1, v_2 \leftarrow x_1][w_1, w_2 \leftarrow x_2]$$

Finally, we distribute the μ-abstraction to each tuple:

$$(a_1)a_2[a_1, a_2 \twoheadleftarrow \mu\gamma.\langle\, (v_1)w_1\gamma_1, (v_2)w_2\gamma_2\,\rangle[\gamma_1, \gamma_2 \leftarrow \gamma]][v_1, v_2 \leftarrow x_1][w_1, w_2 \leftarrow x_2]$$

$$\wr_s$$

$$(((\mu\gamma_1.(v_1)w_1\gamma_1)\mu\gamma_2.(v_2)w_2\gamma_2)[\gamma_1, \gamma_2 \leftarrow \gamma])[v_1, v_2 \leftarrow x_1][w_1, w_2 \leftarrow x_2]$$

The example above shows that applying duplication rules require additional rules such as lifting and congruence rules. We also need compounding and unary sharing rules.

Let $\chi_i, \chi'_j, \chi''_k$ be variables in $\mathbb{T} \cup \mathbb{S}$, and $\tau \in \mathbb{T} \cup \mathbb{S}$ be a term or a stream. Recall that we denote by $@(t, \tau)$ applications $(t)\tau, t \circ \tau$ and by $\mathcal{A}x.t$ abstractions $\lambda x.t, \mu\alpha.t\{\alpha/x\}$. Wherever there are unary sharings, we apply the unary rule instead of the others. To simplify, in many cases we write $[\Phi] \longrightarrow_s \{\ldots\}[\Psi]$ instead of $u^*[\Phi] \longrightarrow_s u^*\{\ldots\}[\Psi]$. All reduction rules are shown below:

Congruence rule

1. $[\phi][\psi] \sim [\psi][\phi]$ when $[\psi]$ does not bind in $[\phi]$

Lifting rules

Forbidden for $p = 1$.

1. $\mathcal{A}x.(u[\phi]) \longrightarrow_s (\mathcal{A}x.u)[\phi]$ if $x \in FV(u)$
2. $@(u[\phi], \tau) \longrightarrow_s @(u, \tau)[\phi]$
3. $[\vec{\chi_p} \leftarrow \tau[\phi]] \longrightarrow_s [\vec{\chi_p} \leftarrow \tau][\phi]$
4. $[\vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p[\phi]] \longrightarrow_s [\vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p][\phi]$ if $y \in FV(t^p)$

Compounding rules

Forbidden for $p = 1$, and for $m = n = 0$.

1. $[\vec{\chi_p} \leftarrow \chi][\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau] \longrightarrow_s [\vec{\chi'_m}, \vec{\chi_p}, \vec{\chi''_n} \leftarrow \tau]$

Unary sharing rules

1. $[\chi \leftarrow \tau] \longrightarrow_s \{\tau/\chi\}$

Duplication rules

Forbidden for $p = 1$.

1. $[\vec{\chi_p} \leftarrow @(v, \tau)] \longrightarrow_s \{@(\chi'_i, \chi''_i)/\chi_i\}[\vec{\chi'_p} \leftarrow v][\vec{\chi''_p} \leftarrow \tau]$
2. $[\vec{x_p} \leftarrow \mathcal{A}x.t] \longrightarrow_s [\vec{x_p} \twoheadleftarrow \mathcal{A}x.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t]]$
3. $[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\vec{z_q} \leftarrow y]] \longrightarrow_s \{\, \mathcal{A}y_i.t_i[\vec{z_i}^{l_i} \leftarrow y_i] \,/x_i\}$
   and $\{z_i^1, \ldots, z_i^{l_i}\} = \{\vec{z_q}\} \cap FV(t_i)$

μ, β rules

1. $(\lambda x.t)u \longrightarrow_{\beta_t} t\{u/x\}$
2. $(\lambda x.t)(u \circ S) \longrightarrow_{\beta_s} (t\{u/x\})S$
3. $(\mu\beta.t)u \longrightarrow_{\mu_t} \mu\beta.t\{(u \circ \beta)/\beta\}$
4. $(\mu\beta.t)S \longrightarrow_{\mu_s} t\{S/\beta\}$

Remark 2.2.5. We often use $\longrightarrow_{\beta,\mu}$ to say that we refer to one of these rules.

2.3 Typing terms in the sequent calculus

We present two simple type systems for $\Lambda\mu S_a$, one in natural deduction (sequent-style), and another in open deduction. In these systems, we have conjunctive types $A_1 \wedge \cdots \wedge A_p$ corresponding to tuples t^p, and negated types $\neg A$ corresponding to μ-variables and streams.

The $\Lambda\mu$-calculus was introduced as an extension of the $\lambda\mu$-calculus that would recover Böhm's separation theorem, which states that for two distinct normal forms t, u, there is a context $\mathcal{C}[-]$ such that $\mathcal{C}[t]$ and $\mathcal{C}[u]$ reduce to two chosen different terms. While our syntax extends Saurin's $\Lambda\mu$-calculus, our type system is more limited. Terms under a μ-abstraction are typed with $\bot$, thus terms needed to prove separation are not typable in our system, which is closer to Parigot's type system.

The sequent system extends classical natural deduction. It features explicit contractions corresponding to all linear occurrences of a $\Lambda\mu S_a$-expression, and closures are constructed with cuts. A term can be typed with $\bot$, an atom a, or an implication $A \to B$.

2.3.1 Atomic $\Lambda\mu$-abstractions

$$\frac{t : \Gamma, A^x \vdash B \mid \Delta}{\lambda x.t : \Gamma \vdash A \to B \mid \Delta}\,\lambda \qquad \frac{t : \Gamma \vdash \bot \mid A^\alpha, \Delta}{\mu\alpha.t : \Gamma \vdash A \mid \Delta}\,\mu$$

2.3.2 Atomic $\Lambda\mu$-applications

$$\frac{t : \Gamma \vdash A \to B \mid \Delta \qquad u : \Gamma' \vdash A \mid \Delta'}{(t)u : \Gamma, \Gamma' \vdash B \mid \Delta, \Delta'}\,@$$

$$\frac{t : \Gamma \vdash B \mid \Delta \qquad S : \Gamma' \vdash \neg B \mid \Delta'}{(t)S : \Gamma, \Gamma' \vdash \bot \mid \Delta, \Delta'}\,@_n$$

$$\frac{t : \Gamma \vdash B \mid \Delta \qquad S : \Gamma' \vdash \neg A \mid \Delta'}{(t \circ S) : \Gamma, \Gamma' \vdash \neg(B \to A) \mid \Delta, \Delta'}\,\circ$$

2.3.3 Atomic $\Lambda\mu$-variables

$$\frac{}{x : A^x \vdash A}\,\text{Var} \qquad \frac{}{\alpha : \vdash \neg A \mid A^\alpha}\,\mu\text{-Var}$$

2.3.4 Atomic $\Lambda\mu$-tuples

$$\frac{t_1 : \Gamma_1 \vdash A_1 \mid \Delta_1 \qquad \dots \qquad t_k : \Gamma_k \vdash A_k \mid \Delta_k}{\langle\, t_1, \dots, t_k \,\rangle : \Gamma_1, \dots, \Gamma_k \vdash A_1 \wedge \dots \wedge A_k \mid \Delta_1, \dots \Delta_k}\ \langle\ \rangle_k$$

2.3.5 Atomic $\Lambda\mu$-sharings

$$\frac{t^* : \Gamma, A^{x_1}, \dots, A^{x_q} \vdash B \mid \Delta \qquad u : \Gamma' \vdash A \mid \Delta'}{t^*[x_1, \dots, x_q \leftarrow u] : \Gamma, \Gamma' \vdash B \mid \Delta, \Delta'}\ \leftarrow$$

$$\frac{t^* : \Gamma \vdash B \mid A^{\alpha_1}, \dots, A^{\alpha_q}, \Delta \qquad S : \Gamma' \vdash \neg A \mid \Delta'}{t^*[\alpha_1, \dots, \alpha_q \leftarrow S] : \Gamma, \Gamma' \vdash B \mid \Delta, \Delta'}\ \leftarrow'$$

2.3.6 Atomic $\Lambda\mu$-distributors

$$\frac{t^* : \Gamma, (A \to B_1)^{x_1}, \dots, (A \to B_q)^{x_q} \vdash C \mid \Delta \qquad u^q : \Gamma', A \vdash B_1 \wedge \dots \wedge B_q \mid \Delta'}{t^*[x_1, \dots, x_q \twoheadleftarrow \lambda y.u^q] : \Gamma, \Gamma' \vdash C \mid \Delta, \Delta'}\ \twoheadleftarrow$$

$$\frac{t^* : \Gamma, A^{x_1}, \dots, A^{x_q} \vdash B \mid \Delta \qquad u^q : \Gamma' \vdash \underbrace{\bot \wedge \dots \wedge \bot}_{q} \mid A^{\alpha}, \Delta'}{t^*[x_1, \dots, x_q \twoheadleftarrow \mu\alpha.u^q] : \Gamma, \Gamma' \vdash B \mid \Delta, \Delta'}\ \twoheadleftarrow'$$

2.4 Typing terms in Open Deduction

In the previous section, we considered a natural deduction system to type atomic terms. In explicit substitution calculi, typing is also done via variants of natural deduction systems, and in this setting, scope (closure under an abstraction), (co-)contraction and weakening are implicit. Also, constructions such as sharings involve cuts, making reduction rules (cut-elimination) unnatural. Defining corresponding reduction rules is one of the first steps to define the calculus, and for each additional rule, one needs to check that soundness is preserved. Furthermore, there is usually no guidance on the best way to define these rules. However in open deduction, they are made structurally explicit, giving constraints on reduction rules to ensure soundness, and making sharing construction immediate. The correspondence between open deduction and the syntax is then natural. From open deduction and its explicit (co-)contraction and scope, we get the medial rules, which are at the core of the atomic $\Lambda\mu S$-calculus. One can note that it is possible to embed an explicit substitution calculus in open deduction, therefore this proof system could give a guidance to construct a calculus with explicit substitutions. However, it is uncertain whether following such a method is efficient.

The open deduction formalism allows us to write two dimensional proofs: horizontal (with connectives) corresponding categorically to the functoriality of connectives, and vertical composition of derivations corresponding to categorical composition.

Derivations in open deduction are constructed as follows:

Definition 2.4.1.

$$\begin{array}{c} A \\ \Downarrow \\ C \end{array} ::= A \mid \frac{A}{C}\mathsf{r} \mid \begin{array}{c} A_1 \\ \Downarrow \\ C_1 \end{array} \wedge \begin{array}{c} A_2 \\ \Downarrow \\ C_2 \end{array} \mid \begin{array}{c} A_1 \\ \Downarrow \\ C_1 \end{array} \vee \begin{array}{c} A_2 \\ \Downarrow \\ C_2 \end{array} \mid \begin{array}{c} C_1 \\ \Uparrow \\ A_1 \end{array} \to \begin{array}{c} A_2 \\ \Downarrow \\ C_2 \end{array} \mid \begin{array}{c} A \\ \Downarrow \\ \frac{B}{B} \\ \Downarrow \\ C \end{array}$$

Arrows indicate a derivation that goes from the premise formula A to the conclusion formula C. After composing derivations, we obtain a derivation from the top formula to the bottom formula. In particular, an implication gives a derivation from $A = C_1 \to A_2$ to $C = A_1 \to C_2$.

We write $\frac{A}{C}\mathsf{r}$ when applying an inference rule r, and double inferences $\frac{A}{\frac{}{A'}}$ correspond to invertible rules for associativity, commutativity and units.

As described in section 1.2.4, the main innovation that comes with open deduction is atomicity. Atomicity for μ-astractions is possible with another medial rule:

$$\frac{(A \wedge B) \vee C}{(A \vee C) \wedge (B \vee C)}\,\mathtt{m'}$$

Instead of duplicating a μ-abstraction $\mu\alpha.t$, we can duplicate t independently, then distribute $\mu\alpha$. If $t : \bot$ and $\alpha : C$:

$$\frac{\mu\alpha.t : \bot \vee C^\alpha}{\mu\alpha.t : (\bot \vee C^\alpha) \wedge \mu\alpha.t : (\bot \vee C^\alpha)}\,\triangle \rightsquigarrow \frac{\dfrac{t : \bot}{t : \bot \wedge t : \bot}\,\triangle \vee C^\alpha}{\mu\alpha.t : (\bot \vee C^\alpha) \wedge \mu\alpha.t : (\bot \vee C^\alpha)}\,\mathtt{m'}$$

2.4.1 Inference rules

In addition to the rules for the atomic λ-calculus, we need to consider corresponding rules for μ-abstraction, stream application and stream construction.

As we type λ-variables and λ-terms with formulas A, dually μ-variables and streams can be typed with negated formulas $\neg A$.

As in the atomic λ-calculus, whenever possible, we would like to avoid disjunctions,

have conjunctive formulas Γ as premises, and have formulas $A \to B$, atoms or $\bot$ as main conclusions. Thus distributivity (medial) and co-contraction are used to define distributors, but are never explicitly used. We omit disjunctions denoting μ-variables until a μ-variable becomes bound with a μ-abstraction.

Equality rules

We use double inferences $\dfrac{A}{\overline{\overline{B}}}$ to denote an equation between A and B. In particular, we use the following equations:

$$\frac{A}{\overline{\overline{A}}} \qquad \frac{\bot \vee A}{\overline{\overline{A}}}$$

Abstraction rules

The μ-abstraction rule is very similar to the λ-abstraction rule. In $\mu\alpha.t$, the μ-variable $\alpha : \neg A$ must occur free in t, just as in $\lambda x.t$ the λ-variable $x : A$ must occur free. Then, whereas λ-abstraction corresponds to combining the derivation of t with the implication $A \to$, μ-abstraction corresponds to the disjunction of the derivation of t with A.

$$\frac{B}{A \to (A \wedge B)}\,\lambda \qquad \frac{B}{(B \wedge \neg A) \vee A}\,\mu$$

Application rules

The three applications correspond to different forms of conjunctions. Term application is followed by modus ponens, and stream application is followed by $\bot$ introduction. Streams $t \circ S$ are constructed by adding a term $t : B$ to a stream $S : \neg A$. The type of $t \circ S$ can therefore be seen as $B \wedge \neg A$, which is classically equivalent to the negated type $\neg(B \to A)$. We then obtain negated types for μ-variables and streams.

$$\frac{A \wedge (A \to B)}{B}\,@ \qquad \frac{A \wedge \neg A}{\bot}\,@_{\mathsf{n}} \qquad \frac{B \wedge \neg A}{\neg(B \to A)}\,\circ$$

Contraction rules

$$\frac{A}{A \wedge \cdots \wedge A}\,\triangle \qquad \frac{A \vee \cdots \vee A}{A}\,\triangledown$$

Distributivity rules

$$\frac{(A_1 \vee \cdots \vee A_n) \to (B_1 \wedge \cdots \wedge B_n)}{(A_1 \to B_1) \wedge \cdots \wedge (A_n \to B_n)} \qquad \frac{(A_1 \wedge \cdots \wedge A_n) \vee C}{(A_1 \vee C) \wedge \cdots \wedge (A_n \vee C)}\,\delta$$

Distributors

$$\frac{A \to (B_1 \wedge \cdots \wedge B_n)}{(A \to B_1) \wedge \cdots \wedge (A \to B_n)}\,\mathbf{d}_\lambda \qquad \frac{(\bot \wedge \cdots \wedge \bot) \vee B}{B \wedge \cdots \wedge B}\,\mathbf{d}_\mu$$

2.4.2 $\Lambda\mu S_a$-terms

Typing derivations for $\Lambda\mu S_a$-terms are as follows:

$$x \equiv A^x \qquad \alpha \equiv \neg A$$

$$\lambda x.t \equiv \dfrac{\dfrac{\Gamma}{\Gamma \wedge A^x}\,\lambda}{A^x \to \overset{\mathsf{t}}{\Big\Downarrow} C} \qquad \mu\alpha.t \equiv \dfrac{\dfrac{\dfrac{\Gamma}{\Gamma \wedge \neg A}\,\mu}{\overset{\mathsf{t}}{\Big\Downarrow} \vee A}}{\dfrac{\bot}{\dfrac{\bot}{A}}}$$

$$\equiv \dfrac{\dfrac{\Gamma}{\Gamma \wedge A^x}\,\lambda}{\dfrac{A^x}{A^x} \to \overset{\mathsf{t}}{\Big\Downarrow} C} \qquad \equiv \dfrac{\dfrac{\dfrac{\Gamma}{\Gamma \wedge \neg A}\,\mu}{\overset{\mathsf{t}}{\Big\Downarrow} \vee \dfrac{A}{A}}}{\dfrac{\bot}{\dfrac{\bot \vee A}{A}}}$$

$$(t)u \equiv \dfrac{\overset{\Gamma_t}{\overset{\mathsf{t}}{\Big\Downarrow}} \quad \wedge \quad \overset{\Gamma_u}{\overset{\mathsf{u}}{\Big\Downarrow}}}{A \to B \quad A}\,@ \qquad (t)S \equiv \dfrac{\overset{\Gamma_t}{\overset{\mathsf{t}}{\Big\Downarrow}} \wedge \overset{\Delta_S}{\overset{\mathsf{S}}{\Big\Downarrow}}}{A \quad \neg A}\,@_{\mathsf{n}}$$

$$t \circ S \equiv \dfrac{\overset{\Gamma_t}{\overset{\mathsf{t}}{\Big\Downarrow}} \wedge \overset{\Delta_S}{\overset{\mathsf{S}}{\Big\Downarrow}}}{B \quad \neg A}\,\circ \qquad \langle t_1, \ldots, t_p \rangle \equiv \overset{\Gamma_1}{\overset{\mathsf{t}_1}{\Big\Downarrow}} \wedge \cdots \wedge \overset{\Gamma_n}{\overset{\mathsf{t}_n}{\Big\Downarrow}}$$

$$u^*[\vec{x_n} \leftarrow t] \equiv \dfrac{\dfrac{\overset{\Gamma}{\overset{\mathsf{t}}{\Big\Downarrow}} \wedge \Sigma_{u^*}}{A}}{\dfrac{A \wedge \cdots \wedge A}{A \wedge \cdots \wedge A \wedge \Sigma_{u^*}}}\,\triangle \qquad u^*[\vec{\gamma_n} \leftarrow S] \equiv \dfrac{\dfrac{\overset{\Delta}{\overset{\mathsf{S}}{\Big\Downarrow}} \wedge \Sigma_{u^*}}{\neg A}}{\dfrac{\neg A \wedge \cdots \wedge \neg A}{\neg A \wedge \cdots \wedge \neg A \wedge \Sigma_{u^*}}}\,\triangle$$

$$\overset{\mathsf{u}^*}{\Big\Downarrow} C \qquad \overset{\mathsf{u}^*}{\Big\Downarrow} C$$

$$u^*[\vec{x_n} \twoheadleftarrow \lambda y.t^n] \quad \equiv \quad \dfrac{\dfrac{\dfrac{\Gamma}{\Gamma \wedge A^y}\,\lambda}{A^y \to \quad \mathsf{t}^{\mathsf{p}}\Big\Downarrow \qquad\qquad \wedge \Sigma_u^*}}{\dfrac{B_1 \wedge \cdots \wedge B_n}{(A \to B_1) \wedge \cdots \wedge (A \to B_n)}\,\mathrm{d}_\lambda}}{(A \to B_1) \wedge \cdots \wedge (A \to B_n) \wedge \Sigma_u^*}$$
$$\mathsf{u}^*\Big\Downarrow$$
$$C$$

$$u^*[\vec{x_n} \twoheadleftarrow \mu\alpha.t^n] \quad \equiv \quad \dfrac{\dfrac{\dfrac{\Gamma}{\Gamma \wedge \neg A}\,\mu}{\mathsf{t}^{\mathsf{p}}\Big\Downarrow \qquad \vee A \quad \wedge \Sigma_u^*}}{\dfrac{\bot \wedge \cdots \wedge \bot}{A \wedge \cdots \wedge A}\,\mathrm{d}_\mu}}{A \wedge \cdots \wedge A \wedge \Sigma_u^*}$$
$$\mathsf{u}^*\Big\Downarrow$$
$$C$$

An alternative presentation in open deduction can be done with boxes. Here is an example with atomic λ-terms:

$$x \equiv \boxed{A^x} \qquad \langle t_1, \ldots, t_n \rangle \equiv \boxed{\begin{array}{c}\Gamma_1 \\ t_1 \| \\ B_1\end{array}} \wedge \ \ldots \ \wedge \boxed{\begin{array}{c}\Gamma_n \\ t_n \| \\ B_n\end{array}} \qquad (t)u \equiv \boxed{\begin{array}{c}\boxed{\begin{array}{c}\Gamma \\ t \| \\ A \to B\end{array}} \wedge \boxed{\begin{array}{c}\Delta \\ u \| \\ A\end{array}} \\ \hdotsfor{1} \\ @\ \dfrac{A \to B \wedge A}{B}\end{array}}$$

$$\lambda x.t \equiv \begin{array}{c}\boxed{\lambda\ \dfrac{\Gamma}{A \to \Gamma \wedge A}} \\ \hdotsfor{1} \\ \boxed{A^x} \to \boxed{\begin{array}{c}\Gamma \wedge A^x \\ t \| \\ B\end{array}}\end{array} \qquad u[x_1, \ldots, x_n \leftarrow t] \equiv \boxed{\begin{array}{c}\boxed{\begin{array}{c}\boxed{\begin{array}{c}\Gamma \\ t \| \\ A\end{array}} \\ \hdotsfor{1} \\ \Delta\ \dfrac{A}{A^{x_1} \wedge \cdots \wedge A^{x_n}}\end{array}} \wedge \boxed{\Delta} \\ \hdotsfor{1} \\ \begin{array}{c}A^{x_1} \wedge \cdots \wedge A^{x_n} \wedge \Delta \\ u \| \\ B\end{array}\end{array}}$$

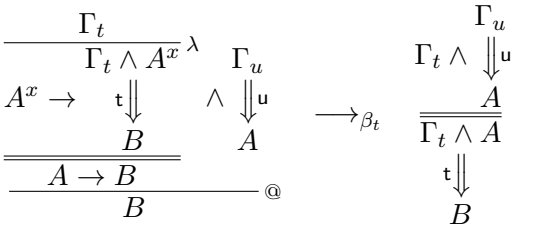
$$u[x_1, \ldots, x_n \twoheadleftarrow \lambda y.t^n] \equiv \boxed{\begin{array}{c}\boxed{\begin{array}{c}\boxed{\lambda\ \dfrac{\Gamma}{A \to \Gamma \wedge A}} \\ \hdotsfor{1} \\ \boxed{A^y} \to \boxed{\begin{array}{c}\Gamma \wedge A^y \\ t^n \| \\ B_1 \wedge \cdots \wedge B_n\end{array}} \wedge \boxed{\Delta} \\ \hdotsfor{1} \\ \mathbf{d}\ \dfrac{A \to B_1 \wedge \cdots \wedge B_n}{A \to B_1 \wedge \cdots \wedge A \to B_n}\end{array}} \\ \hdotsfor{1} \\ \begin{array}{c}A \to B_1 \wedge \cdots \wedge A \to B_n \wedge \Delta \\ u \| \\ C\end{array}\end{array}}$$

2.4.3 Reduction rules

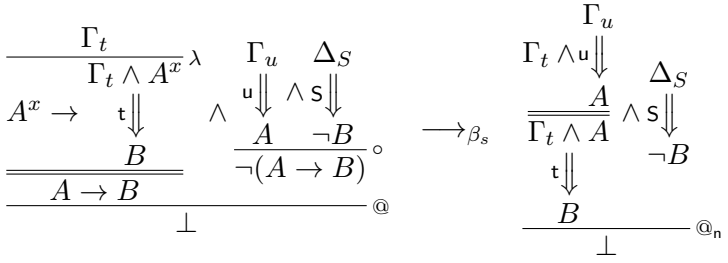
Reduction rules in the atomic $\lambda\mu$-calculus correspond to the following transformations in open deduction, which include cut-elimination:

μ, β rules

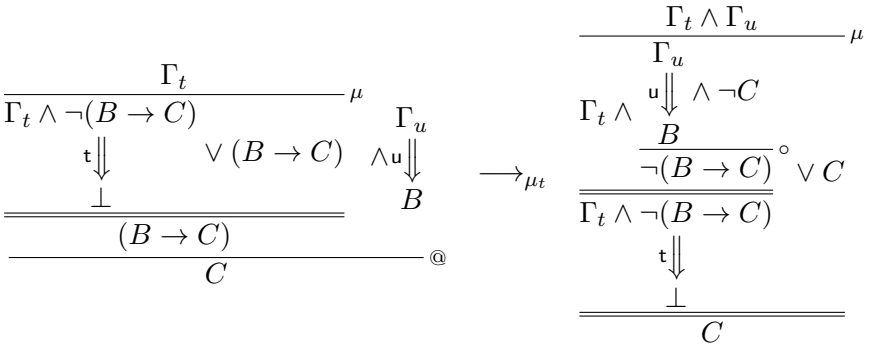
- $(\lambda x.t)u \longrightarrow_{\beta_t} t\{u/x\}$



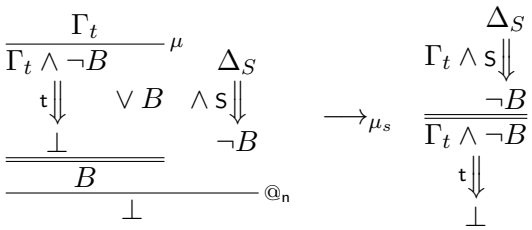
- $(\lambda x.t)(u \circ S) \longrightarrow_{\beta_s} (t\{u/x\})S$



- $(\mu\beta.t)u \longrightarrow_{\mu_t} \mu\beta.t\{u \circ \beta \ / \ \beta\}$

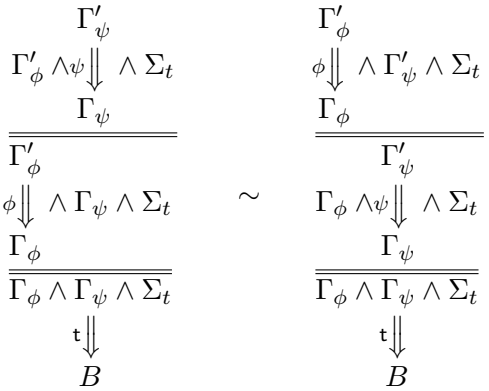


- $(\mu\beta.t)S \longrightarrow_{\mu_s} t\{S/\beta\}$



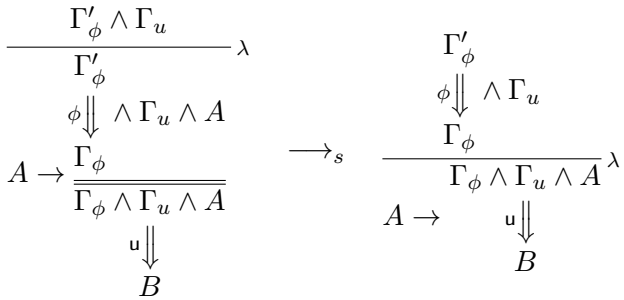
Congruence rules

- $t^*[\phi][\psi] \sim t^*[\psi][\phi]$

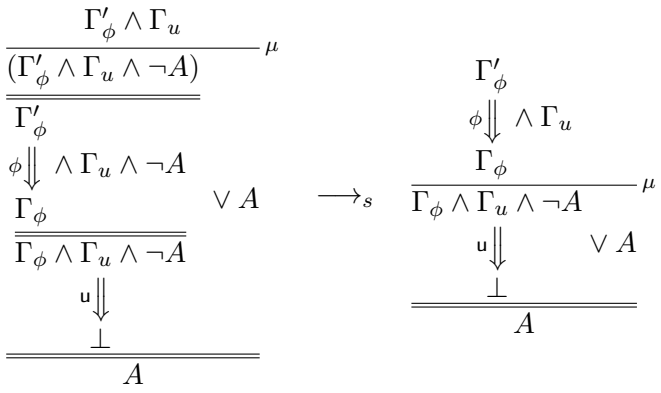


Lifting rules

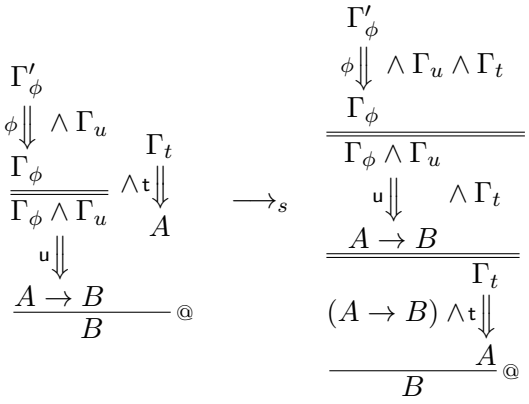
- $\lambda x.(u[\phi]) \longrightarrow_s (\lambda x.u)[\phi]$ if $x \in FV(u)$



- $\mu\alpha.(u[\phi]) \longrightarrow_s (\mu\alpha.u)[\phi]$ if $\alpha \in FV(u)$



- $\begin{matrix}(u[\phi])t \text{ (shown)} \\ (u)t[\phi] \text{ (similar)}\end{matrix} \longrightarrow_s ((u)t)[\phi]$



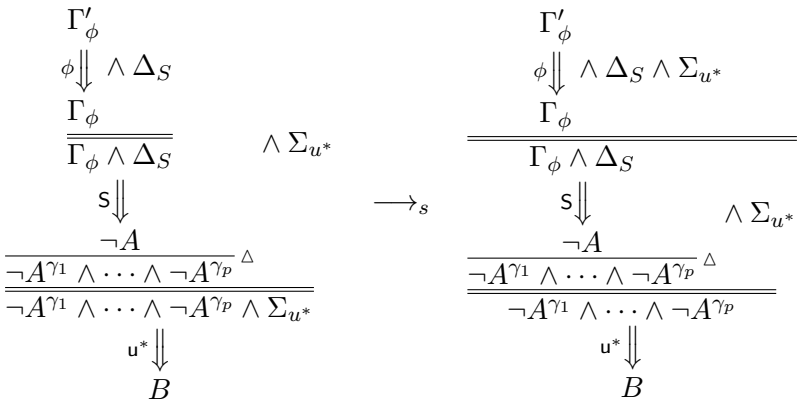
- $$\begin{array}{rcl}(u[\phi])S & \longrightarrow_s & ((u)S)[\phi] \text{ (shown)} \\ (u[\phi]) \circ S & \longrightarrow_s & (u \circ S)[\phi] \text{ (similar)}\end{array}$$

$$\dfrac{\dfrac{\dfrac{\Gamma'_\phi}{\phi\Big\Downarrow \wedge \Gamma_u}}{\dfrac{\Gamma_\phi}{\Gamma_\phi \wedge \Gamma_u}} \wedge \dfrac{\Delta_S}{\mathsf{s}\Big\Downarrow \neg A}}{\dfrac{\mathsf{u}\Big\Downarrow}{\dfrac{A}{\bot}\,@_{\mathsf{n}}}} \quad \longrightarrow_s \quad \dfrac{\dfrac{\dfrac{\Gamma'_\phi}{\phi\Big\Downarrow \wedge \Gamma_u \wedge \Delta_S}}{\dfrac{\Gamma_\phi}{\Gamma_\phi \wedge \Gamma_u}}}{\dfrac{\mathsf{u}\Big\Downarrow \quad \wedge \Delta_S}{\dfrac{\dfrac{A}{\Delta_S}}{\dfrac{A \wedge \mathsf{s}\Big\Downarrow}{\dfrac{\neg A}{\bot}\,@_{\mathsf{n}}}}}}}$$

- $u^*[\vec{x_p} \leftarrow t[\phi]] \longrightarrow_s u^*[\vec{x_p} \leftarrow t][\phi]$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'_\phi}{\phi\Big\Downarrow \wedge \Gamma_t}}{\dfrac{\Gamma_\phi}{\Gamma_\phi \wedge \Gamma_t}} \wedge \Sigma_{u^*}}{\mathsf{t}\Big\Downarrow}}{\dfrac{\dfrac{A}{A^{x_1} \wedge \cdots \wedge A^{x_p}}\,\triangle}{A^{x_1} \wedge \cdots \wedge A^{x_p} \wedge \Sigma_{u^*}}}}{\mathsf{u}^*\Big\Downarrow}}{B} \quad \longrightarrow_s \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'_\phi}{\phi\Big\Downarrow \wedge \Gamma_t \wedge \Sigma_{u^*}}}{\dfrac{\Gamma_\phi}{\Gamma_\phi \wedge \Gamma_t}}}{\mathsf{t}\Big\Downarrow \quad \wedge \Sigma_{u^*}}}{\dfrac{\dfrac{A}{A^{x_1} \wedge \cdots \wedge A^{x_p}}\,\triangle}{A^{x_1} \wedge \cdots \wedge A^{x_p} \wedge \Sigma_{u^*}}}}{\mathsf{u}^*\Big\Downarrow}}{B}$$

- $u^*[\vec{\gamma_q} \leftarrow S[\phi]] \longrightarrow_s u^*[\vec{\gamma_q} \leftarrow S][\phi]$



- $u^*[\vec{x_p} \twoheadleftarrow \lambda y.t^p[\phi]] \longrightarrow_s u^*[\vec{x_p} \twoheadleftarrow \lambda y.t^p][\phi] \qquad \text{if } y \in FV(t^p)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_{t^p} \wedge \Gamma'_\phi}{\Gamma'_\phi}\lambda}{\phi\Big\Downarrow \wedge \Gamma_{t^p} \wedge A^y} \quad A^y \to \dfrac{\Gamma_\phi}{\Gamma_\phi \wedge \Gamma_{t^p} \wedge A^y} \quad \wedge \Sigma_{u^*}}{\mathsf{t}^\mathsf{p}\Big\Downarrow}}{B_1 \wedge \cdots \wedge B_p}}{A \to (B_1 \wedge \cdots \wedge B_p)}}{(A \to B_1) \wedge \cdots \wedge (A \to B_p)}\mathbf{d}_\lambda}{(A \to B_1) \wedge \cdots \wedge (A \to B_p) \wedge \Sigma_{u^*}}}{\mathsf{u}^*\Big\Downarrow}}{B}$$

$\longrightarrow_s$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'_\phi}{\phi\Big\Downarrow \wedge \Gamma_{t^p} \wedge \Sigma_{u^*}}}{\Gamma_\phi}}{\dfrac{\Gamma_{t^p} \wedge \Gamma_\phi}{\Gamma_\phi \wedge \Gamma_{t^p} \wedge A^y}\lambda}}{A^y \to \quad \mathsf{t}^\mathsf{p}\Big\Downarrow \quad \wedge \Sigma_{u^*}}}{B_1 \wedge \cdots \wedge B_p}}{(A \to B_1) \wedge \cdots \wedge (A \to B_p)}\mathbf{d}_\lambda}{(A \to B_1) \wedge \cdots \wedge (A \to B_p) \wedge \Sigma_{u^*}}}{\mathsf{u}^*\Big\Downarrow}}{B}$$

- $u^*[\vec{x_p} \leftarrow \mu\alpha.t^p[\phi]] \longrightarrow_s u^*[\vec{x_p} \leftarrow \mu\alpha.t^p][\phi] \qquad \text{if } \alpha \in FV(t^p)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_{t^p} \wedge \Gamma'_\phi}{\Gamma'_\phi}\mu}{\phi\Big\Downarrow \wedge \Gamma_{t^p} \wedge \neg A}}{\dfrac{\Gamma_\phi}{(\Gamma_\phi \wedge \Gamma_{t^p} \wedge \neg A)}} \vee A \quad \wedge \Sigma_{u^*}}{\mathsf{t}^p\Big\Downarrow}}{\dfrac{\bot \wedge \cdots \wedge \bot}{(\ A \wedge \cdots \wedge A\ )}\mathrm{d}_\mu}}{(\ A \wedge \cdots \wedge A\ ) \wedge \Sigma_{u^*}}}{\mathsf{u}^*\Big\Downarrow}}{B} \longrightarrow_s \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'_\phi}{\phi\Big\Downarrow \wedge \Gamma_{t^p}}}{\dfrac{\Gamma_\phi}{(\Gamma_\phi \wedge \Gamma_{t^p} \wedge \neg A)}}\mu \quad \wedge \Sigma_{u^*}}{\mathsf{t}^p\Big\Downarrow \qquad \vee A}}{\dfrac{\bot \wedge \cdots \wedge \bot}{(\ A \wedge \cdots \wedge A\ )}\mathrm{d}_\mu}}{(\ A \wedge \cdots \wedge A\ ) \wedge \Sigma_{u^*}}}{\mathsf{u}^*\Big\Downarrow}}{B}$$

Compounding rules

- $u^*[\vec{y_q} \leftarrow y][\vec{x_p}, y, \vec{z_r} \leftarrow t] \longrightarrow_s u^*[\vec{x_p}, \vec{y_q}, \vec{z_r} \leftarrow t]$

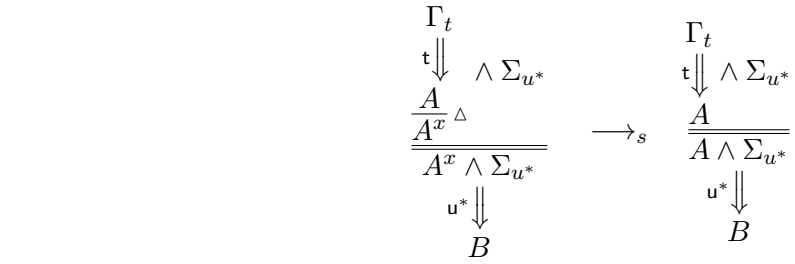
$$\dfrac{\dfrac{\Gamma^{\vec{x}} \wedge \dfrac{\dfrac{\dfrac{\Gamma_t}{\mathsf{t}\Big\Downarrow}}{B}}{B^{y_1} \wedge \cdots \wedge B^{y_q}}\triangle \wedge \Gamma^{\vec{z}}}{\triangle} \quad \wedge \Sigma_{u^*}}{\Gamma^{\vec{x}} \wedge B^{y_1} \wedge \cdots \wedge B^{y_q} \wedge \Gamma^{\vec{z}} \wedge \Sigma_{u^*}}}{\mathsf{u}^*\Big\Downarrow}}{C} \longrightarrow_s \dfrac{\dfrac{\dfrac{\dfrac{\Gamma_t}{\mathsf{t}\Big\Downarrow}}{B}}{\Gamma^{\vec{x}} \wedge B^{y_1} \wedge \cdots \wedge B^{y_q} \wedge \Gamma^{\vec{z}}}\triangle \quad \wedge \Sigma_{u^*}}{\Gamma^{\vec{x}} \wedge B^{y_1} \wedge \cdots \wedge B^{y_q} \wedge \Gamma^{\vec{z}} \wedge \Sigma_{u^*}}}{\mathsf{u}^*\Big\Downarrow}}{C}$$

where $\Gamma^{\vec{x}} = B^{x_1} \wedge \cdots \wedge B^{x_p}$ and $\Gamma^{\vec{z}} = B^{z_1} \wedge \cdots \wedge B^{z_r}$

- $u^*[\vec{\alpha_q} \leftarrow \gamma][\vec{\beta_p}, \gamma, \vec{\delta_r} \leftarrow S] \longrightarrow_s u^*[\vec{\beta_p}, \vec{\alpha_q}, \vec{\delta_r} \leftarrow S]$: similar

Unary sharing rules

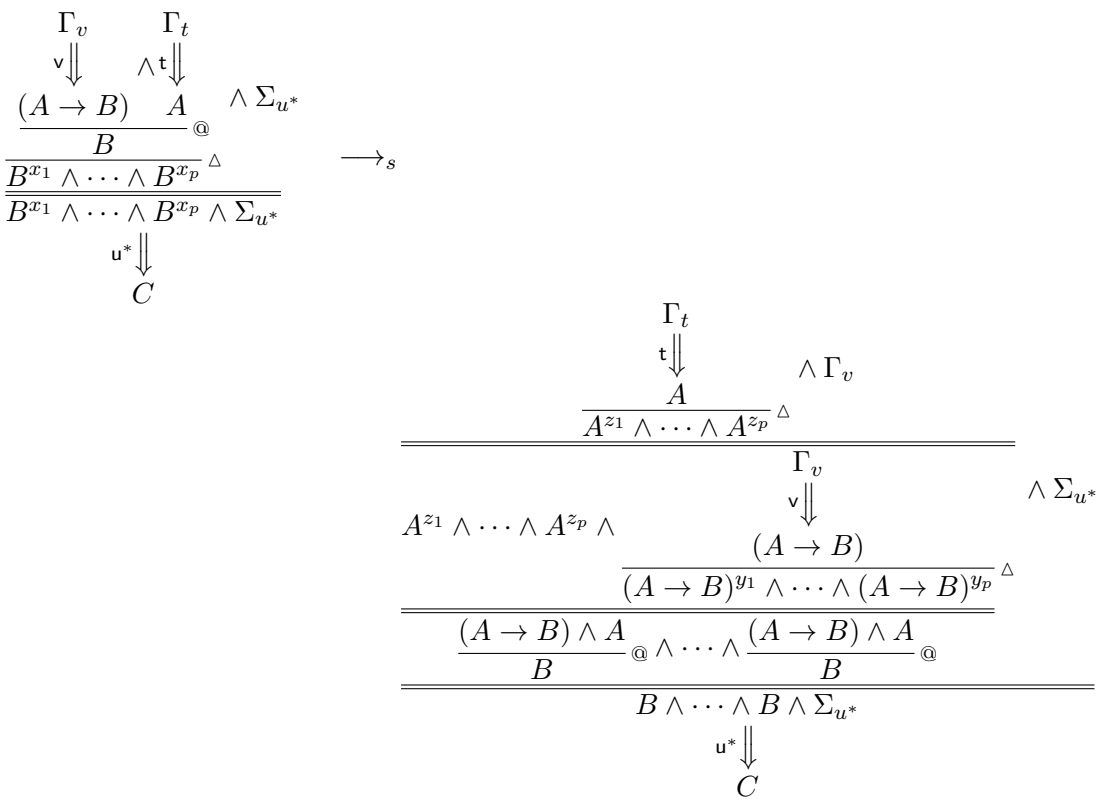
- $u^*[x \leftarrow t] \longrightarrow_s u^*\{t/x\}$



- $u^*[\alpha \leftarrow S] \longrightarrow_s u^*\{S/\alpha\}$: similar

Duplication rules

- $u^*[\vec{x_p} \leftarrow (v)t] \longrightarrow_s u^*\{(y_i)z_i/x_i\}[\vec{y_p} \leftarrow v][\vec{z_p} \leftarrow t]$



- $u^*[\vec{\alpha_p} \leftarrow t \circ S'] \longrightarrow_s u^*\{y_i \circ \alpha'_i \ / \ \alpha_i\}[\vec{y_p} \leftarrow t][\vec{\alpha'_p} \leftarrow S']$: similar

- $u^*[\vec{x_p} \leftarrow \lambda x.t] \longrightarrow_s u^*[\vec{x_p} \twoheadleftarrow \lambda x.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t]]$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_t}{\Gamma_t \wedge A}\,\lambda}{A^x \to \overset{\mathsf{t}}{\Big\Downarrow} B}\ \wedge \Sigma_{u^*}}{(A \to B) \wedge \cdots \wedge (A \to B)}\,\triangle}{(A \to B) \wedge \cdots \wedge (A \to B) \wedge \Sigma_{u^*}} \qquad \longrightarrow_s$$

$$\overset{\mathsf{u}^*}{\Big\Downarrow} C$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_t}{\Gamma_t \wedge A}\,\lambda}{A^x \to \overset{\mathsf{t}}{\Big\Downarrow} B}\ \wedge \Sigma_{u^*}}{B \wedge \cdots \wedge B}\,\triangle}{(A \to B) \wedge \cdots \wedge (A \to B)}\,\mathrm{d}_\lambda}{(A \to B) \wedge \cdots \wedge (A \to B) \wedge \Sigma_{u^*}}$$

$$\overset{\mathsf{u}^*}{\Big\Downarrow} C$$

- $u^*[\vec{x_p} \leftarrow \mu\alpha.t] \longrightarrow_s u^*[\vec{x_p} \twoheadleftarrow \mu\alpha.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t]]$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_t}{\Gamma_t \wedge \neg A}\,\mu}{\overset{\mathsf{t}}{\Big\Downarrow} \vee A}\ \wedge \Sigma_{u^*}}{\bot}}{A}\,\triangle}{A^{x_1} \wedge \cdots \wedge A^{x_p}}\,\triangle}{A^{x_1} \wedge \cdots \wedge A^{x_p} \wedge \Sigma_{u^*}} \qquad \longrightarrow_s$$

$$\overset{\mathsf{u}^*}{\Big\Downarrow} C$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_t}{\Gamma_t \wedge \neg A}\,\mu}{\overset{\mathsf{t}}{\Big\Downarrow} \vee A}\ \wedge \Sigma_{u^*}}{\bot}}{\bot \wedge \cdots \wedge \bot}\,\triangle}{(\, A \wedge \cdots \wedge A \,)}\,\mathrm{d}_\mu}{(\, A \wedge \cdots \wedge A \,) \wedge \Sigma_{u^*}}$$

$$\overset{\mathsf{u}^*}{\Big\Downarrow} C$$

- $u^*[\vec{x_p} \twoheadleftarrow \lambda y.\langle\, \vec{t_p} \,\rangle[\vec{z_q} \leftarrow y]] \longrightarrow_s u^*\{\lambda y_i.t_i[\vec{z_i}^{l_i} \leftarrow y_i] \,/\, x_i\}$

$$\dfrac{\dfrac{\dfrac{\Gamma_{\vec{t_p}}}{\Gamma_{\vec{t_p}} \wedge \dfrac{A^y}{A^{z_1} \wedge \cdots \wedge A^{z_p}}\,\triangle}}{\Gamma_{\vec{t_p}} \wedge A^{z_1} \wedge \cdots \wedge A^{z_p}}\,\lambda}{A^y \to \quad \Downarrow \mathsf{t^p}} \wedge \Sigma_{u^*}$$

$$\dfrac{B_1 \wedge \cdots \wedge B_p}{(A \to B_1) \wedge \cdots \wedge (A \to B_p)}\,\mathbf{d}_\mu$$

$$\dfrac{}{(A \to B_1) \wedge \cdots \wedge (A \to B_p) \wedge \Sigma_{u^*}}$$

$$\Downarrow \mathsf{u^*}$$

$$C$$

$\longrightarrow_s$

$$\dfrac{\dfrac{\Gamma_{t_1}}{\Gamma_{t_1} \wedge A^{y_1}}\,\lambda}{A^{y_1} \to \quad \Downarrow \mathsf{t_1}} \wedge \cdots \wedge \dfrac{\dfrac{\Gamma_{t_p}}{\Gamma_{t_p} \wedge A^{y_p}}\,\lambda}{A^{y_p} \to \quad \Downarrow \mathsf{t_p}} \wedge \Sigma_{u^*}$$

$$\dfrac{B_1 \qquad\qquad B_p}{(A \to B_1) \wedge \cdots \wedge (A \to B_p) \wedge \Sigma_{u^*}}$$

$$\Downarrow \mathsf{u^*}$$

$$C$$

where $\{z_i^1, \ldots, z_i^{l_i}\} = \{\vec{z_q}\} \cap FV(t_i)$

- $u^*[\vec{x_p} \twoheadleftarrow \mu\alpha.\langle\, \vec{t_p} \,\rangle[\vec{\gamma_q} \leftarrow \alpha]] \longrightarrow_s u^*\{\mu\alpha_i.t_i[\vec{\gamma_i}^{l_i} \leftarrow y_i] \,/\, x_i\}$

$$\dfrac{\dfrac{\dfrac{\Gamma_{\vec{t_p}}}{\Gamma_{\vec{t_p}} \wedge \neg A}\,\mu}{\mathsf{t^p}\Big\Downarrow \quad \vee A \quad \wedge \Sigma_{u^*}}}{\dfrac{\dfrac{\bot \wedge \cdots \wedge \bot}{(\, A \wedge \cdots \wedge A \,)}\,\mathrm{d}_\mu}{(\, A \wedge \cdots \wedge A \,) \wedge \Sigma_{u^*}}}}{\mathsf{u^*}\Big\Downarrow C} \qquad \longrightarrow_s$$

$$\dfrac{\dfrac{\dfrac{\Gamma_{t_1}}{\Gamma_{t_1} \wedge \neg A}\,\mu}{\mathsf{t_1}\Big\Downarrow \quad \vee A} \quad \wedge \cdots \wedge \dfrac{\dfrac{\Gamma_{t_p}}{\Gamma_{t_p} \wedge \neg A}\,\mu}{\mathsf{t_p}\Big\Downarrow \quad \vee A} \quad \wedge \Sigma_{u^*}}{\dfrac{\dfrac{\bot}{A} \qquad \dfrac{\bot}{A}}{(\, A \wedge \cdots \wedge A \,) \wedge \Sigma_{u^*}}}}{\mathsf{u^*}\Big\Downarrow C}$$

where $\{\gamma_i^1, \ldots, \gamma_i^{l_i}\} = \{\vec{\gamma_q}\} \cap FV(t_i)$

We can observe that types are preserved during reductions.

Theorem 2.4.2. (Subject reduction) If $t^* \longrightarrow_{\beta,\mu,s} u^*$ and $t^* : \mathcal{T}$, then $u^* : \mathcal{T}$.

Chapter 3

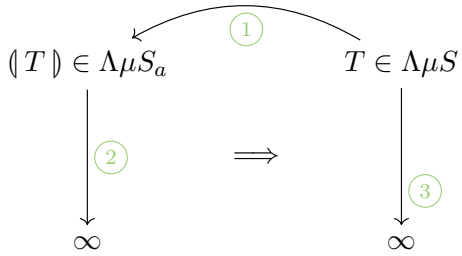
Connecting $\Lambda\mu S_a$ to $\Lambda\mu S$: towards PSN

The subject reduction theorem concludes the previous chapter, ensuring that types are preserved under reduction. Another main property to show is preservation of strong normalization (PSN), stating that if a $\Lambda\mu S$-term is strongly normalizing (i.e. each reduction path terminates), its atomic interpretation in $\Lambda\mu S_a$ is also strongly normalizing. This chapter introduces some core lemmas which attest that our calculus (regarding sharing reductions and translations) is well-behaved, and which are used to prove PSN. While PSN seems like a natural property to expect, it has been shown by Melliès [Mel95] that the pioneer calculus $\lambda\sigma$ with explicit substitutions [ACCL91] did not enjoy PSN. Furthermore, where PSN is valid, it can be challenging to prove. Chapter 4 deals with one main difficulty regarding infinite reductions inside weakenings. Another observation is that sharing reductions correspond to zero steps in the $\Lambda\mu S$-calculus, so we need to show that these steps normalize. We will show a new proof for the strong normalization of sharing reductions in Chapter 5, constructing a measure strictly decreasing with sharing reductions. We finally combine PSN between the atomic calculus and the weakening calculus and PSN for the weakening calculus.

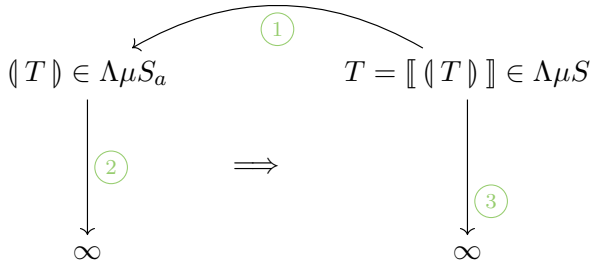
We now give the naive proof idea for PSN and explain some problems that come with them. Recall that for $T \in \Lambda\mu S$, $(\!| T |\!)$ is its translation in the atomic calculus. For $t \in \Lambda\mu S_a$, $[\![t]\!]$ is its interpretation in the $\Lambda\mu S$-calculus. We would like to prove PSN for the atomic calculus with respect to the $\Lambda\mu S$-calculus:

For any term $T \in \Lambda\mu S$, if T is strongly normalizing then its translation $(\!| T |\!) \in \Lambda\mu S_a$ is strongly normalizing.

The easiest way to prove the theorem is to work with its contrapositive: for $T \in \Lambda\mu S$, if $(\!|\, T \,|\!) \in \Lambda\mu S_a$ has an infinite reduction path, then $T \in \Lambda\mu S$ has an infinite reduction path. This will be illustrated as follows: we translate (1) a term $T \in \Lambda\mu S$ into $(\!|\, T \,|\!) \in \Lambda\mu S_a$. If $(\!|\, T \,|\!)$ has an infinite path (2), then T has an infinite path (3).



Conveniently, the lemma below, using $[\![-]\!]$, states that $[\![\, (\!|\, T \,|\!) \,]\!] = T$, i.e. the translation $(\!|\, T \,|\!)$ of a $\Lambda\mu S$-term T can be taken back to T via interpretation $[\![-]\!]$. We can then construct a sequence in $\Lambda\mu S$ from a sequence in $\Lambda\mu S_a$, and describe a direct relation between the atomic calculus and the $\Lambda\mu S$-calculus.



Lemma 3.0.1 (Interpretation is inverse to translation). Let $T \in \Lambda\mu S$, then $[\![\, (\!|\, T \,|\!) \,]\!] = T$.

Proof. By induction on T. Let V^* be a term or a stream, let χ, χ_i be variables. Recall that $\frac{l_i}{\chi_i}$ means replacing χ_i with l_i fresh distinct variables. Let $\sigma = \underbrace{\frac{l_1}{\chi_1} \dots \frac{l_p}{\chi_p}}_{\text{occurrences in } T}$, replacing the different occurrences of each free variable χ_i with fresh, distinct variables. Let $\Phi = [\vec{\chi_1}^{\,l_1} \leftarrow \chi_1] \dots [\vec{\chi_p}^{\,l_p} \leftarrow \chi_p]$. We use the inductive hypothesis: $[\![\, (\!|\, T\sigma \,|\!)' \,]\!] = T\sigma$.

- Let $T = \chi$. Then:

$$[\![\, (\!|\, \chi \,|\!)' \,]\!] = \chi$$

- Let $T = @(U, V^*)$. In this case $\sigma = \sigma_1\sigma_2 = \underbrace{\frac{l_1}{\chi_1}\dots\frac{l_k}{\chi_k}}_{\text{occurrences in } U}\ \underbrace{\frac{l_{k+1}}{\chi_{k+1}}\dots\frac{l_p}{\chi_p}}_{\text{occurrences in } V^*}$. Then:

$$\begin{aligned}
[\![\, (\!| \, @(U, V^*)\sigma \, |\!)' \,]\!] &= [\![\, (\!| \, @(U\sigma_1, V^*\sigma_2) \, |\!)' \,]\!] \\
&= [\![\, @((\!| \, U\sigma_1 \, |\!)', (\!| \, V^*\sigma_2 \, |\!)') \,]\!] \\
&= @([\![\, (\!| \, U\sigma_1 \, |\!)' \,]\!], [\![\, (\!| \, V^*\sigma_2 \, |\!)' \,]\!]) \\
&= @(U\sigma_1, V^*\sigma_2) \\
&= @(U, V^*)\sigma
\end{aligned}$$

- Let $T = \mathcal{A}x.U$. Suppose $|U|_x = p \neq 1$.

  Then:

$$\begin{aligned}
[\![\, (\!| \, (\mathcal{A}x.U)\sigma \, |\!)' \,]\!] &= [\![\, \mathcal{A}x.(\!| \, U\sigma \, |\!)' \,]\!] \\
&= \mathcal{A}x.[\![\, (\!| \, U\sigma \, |\!)' \,]\!] \\
&= \mathcal{A}x.(U\sigma) \\
&= (\mathcal{A}x.U)\sigma
\end{aligned}$$

Then, since $\{\!| \, \Phi \, |\!\}$ and σ are inverse operations, we have

$$[\![\, (\!| \, T \, |\!) \,]\!] = [\![\, (\!| \, T\sigma \, |\!)'\Phi \,]\!] = [\![\, (\!| \, T\sigma \, |\!)' \,]\!]\{\!| \, \Phi \, |\!\} = T\sigma\{\!| \, \Phi \, |\!\} = T$$

□

In particular, PSN would be proven if one could show that one atomic reduction step corresponds to at least one $\Lambda\mu S$-reduction step. Unfortunately this is not true in general, as sharing reductions and reductions inside weakenings correspond to zero steps in the $\Lambda\mu S$-calculus. The lemma below shows that sharing reductions are well-behaved, in the sense that two terms $t \longrightarrow_s^* u$ are equated when translated back in the $\Lambda\mu S$-calculus. That is to say, if two atomic terms are β, μ equivalent and only differ by sharing reductions, they keep the same denotation in the $\Lambda\mu S$-calculus.

Lemma 3.0.2 (Sharing reduction preserves interpretation). For $t, u \in \Lambda\mu S_a$, if $t \longrightarrow_s u$ then $[\![\, t \,]\!] = [\![\, u \,]\!]$.

Proof. • Lifting rules:

1. If $x \in FV(u)$:

$$\begin{aligned}
\mathcal{A}x.(u[\phi]) &\longrightarrow_s (\mathcal{A}x.u)[\phi] \\
[\![\, \mathcal{A}x.(u[\phi]) \,]\!] &= \mathcal{A}x.[\![\, (u[\phi]) \,]\!] \\
&= \mathcal{A}x.([\![\, u \,]\!]\{\!|\, \phi \,|\!\}) \\
&= (\mathcal{A}x.[\![\, u \,]\!])\{\!|\, \phi \,|\!\} \\
&= ([\![\, \mathcal{A}x.u \,]\!])\{\!|\, \phi \,|\!\} \\
&= [\![\, (\mathcal{A}x.u)[\phi] \,]\!]
\end{aligned}$$

2.

$$\begin{aligned}
@(u[\phi], \tau) &\longrightarrow_s @(u, \tau)[\phi] \\
[\![\, @(u[\phi], \tau) \,]\!] &= @([\![\, u[\phi] \,]\!], [\![\, \tau \,]\!]) \\
&= @([\![\, u \,]\!]\{\!|\, \phi \,|\!\}, [\![\, \tau \,]\!]) \\
&= @([\![\, u \,]\!], [\![\, \tau \,]\!])\{\!|\, \phi \,|\!\} \\
&= [\![\, @(u, \tau) \,]\!]\{\!|\, \phi \,|\!\} \\
&= [\![\, @(u, \tau)[\phi] \,]\!]
\end{aligned}$$

3.

$$\begin{aligned}
[\vec{\chi_p} \leftarrow \tau[\phi]] &\longrightarrow_s [\vec{\chi_p} \leftarrow \tau][\phi] \\
\{\!|\, \vec{\chi_p} \leftarrow \tau[\phi] \,|\!\} &= \{[\![\, \tau[\phi] \,]\!]/\chi_i\}_{i \leq p} \\
&= \{[\![\, \tau \,]\!]/\chi_i\}_{i \leq p}\{\!|\, \phi \,|\!\} \\
&= \{\!|\, \vec{\chi_p} \leftarrow \tau \,|\!\}\{\!|\, \phi \,|\!\} \\
&= \{\!|\, [\vec{\chi_p} \leftarrow \tau][\phi] \,|\!\}
\end{aligned}$$

4. If $y \in FV(t^p)$:

$$\begin{aligned}
[\vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p[\phi]] &\longrightarrow_s [\vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p][\phi] \\
\{\!|\, \vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p[\phi] \,|\!\} &= \{\!|\, \vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p \,|\!\}\{\!|\, \phi \,|\!\} \\
&= \{\!|\, [\vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p][\phi] \,|\!\}
\end{aligned}$$

- Compounding rules:

1.

$$
\begin{aligned}
[\vec{\chi_p} \leftarrow \chi][\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau] &\longrightarrow_s [\vec{\chi'_m}, \vec{\chi_p}, \vec{\chi''_n} \leftarrow \tau] \\
\{\!|\, [\vec{\chi_p} \leftarrow \chi][\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau] \,|\!\} &= \{\!|\, \vec{\chi_p} \leftarrow \chi \,|\!\}\{\!|\, \vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau \,|\!\} \\
&= \{\chi/\chi_i\}_{i\leq p}\{[\![\, \tau \,]\!]/\chi'_j\}_{j\leq m} \\
&\quad \{[\![\, \tau \,]\!]/\chi''_k\}_{k\leq n}\{[\![\, \tau \,]\!]/\chi\} \\
&= \{[\![\, \tau \,]\!]/\chi'_j\}_{j\leq m}\{[\![\, \tau \,]\!]/\chi''_k\}_{k\leq n} \\
&\quad \{[\![\, \tau \,]\!]/\chi_i\}_{i\leq p} \\
&= \{\!|\, \vec{\chi'_m}, \vec{\chi_p}, \vec{\chi''_n} \leftarrow \tau \,|\!\}
\end{aligned}
$$

- Unary sharing rules:

  1.

$$
\begin{aligned}
[\chi \leftarrow \tau] &\longrightarrow_s \{\tau/\chi\} \\
\{\!|\, \chi \leftarrow \tau \,|\!\} &= \{[\![\, \tau \,]\!]/\chi\}
\end{aligned}
$$

- Duplication rules:

  1.

$$
\begin{aligned}
[\vec{\chi_p} \leftarrow @(v, \tau)] &\longrightarrow_s \{@(\chi'_i, \chi''_i)/\chi_i\}[\vec{\chi'_p} \leftarrow v][\vec{\chi''_p} \leftarrow \tau] \\
\{\!|\, \vec{\chi_p} \leftarrow @(v, \tau) \,|\!\} &= \{[\![\, @(v, \tau) \,]\!]/\chi_i\}_{i\leq p} \\
&= \{@([\![\, v \,]\!], [\![\, \tau \,]\!])/\chi_i\}_{i\leq p} \\
&= \{@(\chi'_i, \chi''_i)/\chi_i\}_{i\leq p}\{[\![\, v \,]\!]/\chi'_p\}_{i\leq p}\{[\![\, \tau \,]\!]/\chi''_i\}_{i\leq p} \\
&= \{@(\chi'_i, \chi''_i)/\chi_i\}\{\!|\, [\vec{\chi'_p} \leftarrow v][\vec{\chi''_p} \leftarrow \tau] \,|\!\}
\end{aligned}
$$

  2.

$$
\begin{aligned}
[\vec{x_p} \leftarrow \mathcal{A}x.t] &\longrightarrow_s [\vec{x_p} \twoheadleftarrow \mathcal{A}x.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t]] \\
\{\!|\, \vec{x_p} \leftarrow \mathcal{A}x.t \,|\!\} &= \{[\![\, \mathcal{A}x.t \,]\!]/x_i\}_{i\leq p} \\
&= \{\mathcal{A}x.[\![\, t \,]\!]/x_i\}_{i\leq p} \\
&= \{\mathcal{A}x.[\![\, y_i \,]\!]/x_i\}_{i\leq p}\{[\![\, t \,]\!]/y_i\}_{i\leq p} \\
&= \{\!|\, \vec{x_p} \twoheadleftarrow \mathcal{A}x.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t] \,|\!\}
\end{aligned}
$$

3. Let $\{z_i^1, \ldots, z_i^{l_i}\} = \{\vec{z_q}\} \cap FV(t_i)$. Then:

$$\begin{aligned}
[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\vec{z_q} \leftarrow y]] &\longrightarrow_s \{\, \mathcal{A}y_i.t_i[\vec{z_i}^{l_i} \leftarrow y_i] \,/x_i\}_{i \leq p} \\
\{\!\{\, \vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\vec{z_q} \leftarrow y] \,\}\!\} &= \{(\mathcal{A}y.[\![\, t_i \,]\!]\{\!\{\, \vec{z_q} \leftarrow y \,\}\!\})/x_i\}_{i \leq p} \\
&= \{(\mathcal{A}y.[\![\, t_i \,]\!]\{y/z_j\}_{j \leq q})/x_i\}_{i \leq p} \\
&=_\alpha \{(\mathcal{A}y_i.[\![\, t_i \,]\!]\{y_i/z^{l_i}\}_{i \leq p})/x_i\}_{i \leq p} \\
&= \{\, [\![\, \mathcal{A}y_i.t_i[\vec{z_i}^{l_i} \leftarrow y_i] \,]\!] \,/x_i\}_{i \leq p}
\end{aligned}$$

□

Sharing reduction is well-behaved, and we show in Chapter 5 (Theorem 5.5.3) that it is in particular strongly normalizing. Intuitively, this means that procedures such as compounding sharings, evaluating unary sharings, atomically unfolding sharings (duplications), and organizing the term structure (liftings) must terminate. Now if an atomic term t has an infinite reduction path, the path looks like

$$t \longrightarrow_{\beta,\mu} t'_1 \longrightarrow_s^* t_1 \longrightarrow_{\beta,\mu} \ldots$$

with a finite number of s-steps between an infinite number of β or μ-steps.

The following theorem shows that a (β, μ)-step in the atomic calculus is interpreted as zero or more steps in the $\Lambda\mu S$-calculus. Because we have 0-ary closures, an atomic (β, μ)-step can correspond to zero steps in the $\Lambda\mu S$-calculus.

$$\begin{array}{ccc}
t \in \Lambda\mu S_a & \xrightarrow{[\![-]\!]} & [\![\, t \,]\!] \in \Lambda\mu S \\
\Big\downarrow{\scriptstyle 1}_{\beta,\mu} & & \Big\downarrow{\scriptstyle *}_{\beta,\mu} \\
t' \in \Lambda\mu S_a & \xrightarrow{[\![-]\!]} & [\![\, t' \,]\!] \in \Lambda\mu S
\end{array}$$

Theorem 3.0.3. Let $t \in \Lambda\mu S_a$. If $t \longrightarrow_{\beta,\mu} t'$ then $[\![\, t \,]\!] \longrightarrow_{\beta,\mu}^* [\![\, t' \,]\!]$.

Proof. By induction on t.

- If t is a redex:

– If $t = (\lambda x.u)v \longrightarrow_\beta u\{v/x\} = t'$. Then

$$\begin{aligned}[\![\, t \,]\!] &= (\lambda x.[\![\, u \,]\!])[\![\, v \,]\!] \\ &\longrightarrow_\beta [\![\, u \,]\!]\{[\![\, v \,]\!]/x\} \\ &= [\![\, t' \,]\!]\end{aligned}$$

– The other cases are similar.

- If $t = @(u, \tau) \longrightarrow_{\beta,\mu} @(u', \tau) = t'$. By induction hypothesis,

$$[\![\, u \,]\!] \longrightarrow^*_{\beta,\mu} [\![\, u' \,]\!]$$

Therefore

$$[\![\, t \,]\!] = @([\![\, u \,]\!], [\![\, \tau \,]\!]) \longrightarrow^*_{\beta,\mu} @([\![\, u' \,]\!], [\![\, \tau \,]\!]) = [\![\, t' \,]\!]$$

The proof is similar if $t = @(u, \tau) \longrightarrow_{\beta,\mu} @(u, \tau') = t'$.

- If $t = \mathcal{A}x.u \longrightarrow_{\beta,\mu} \mathcal{A}x.u' = t'$. By induction hypothesis,

$$[\![\, u \,]\!] \longrightarrow^*_{\beta,\mu} [\![\, u' \,]\!]$$

Therefore

$$[\![\, t \,]\!] = \mathcal{A}x.[\![\, u \,]\!] \longrightarrow^*_{\beta,\mu} \mathcal{A}x.[\![\, u' \,]\!] = [\![\, t' \,]\!]$$

- If $t^* = u^*[\phi] \longrightarrow_{\beta,\mu} u^{*\prime}[\phi] = t^{*\prime}$. By induction hypothesis, $[\![\, u^* \,]\!] \longrightarrow^*_{\beta,\mu} [\![\, u^{*\prime} \,]\!]$. Therefore:

$$\begin{aligned}[\![\, t^* \,]\!] &= [\![\, u^* \,]\!]\{\!\{\, \phi \,\}\!\} \\ &\longrightarrow^*_{\beta,\mu} [\![\, u^{*\prime} \,]\!]\{\!\{\, \phi \,\}\!\} \\ &= [\![\, t^{*\prime} \,]\!]\end{aligned}$$

- If $t^p = \langle\, u_1, \ldots, u_i, \ldots, u_p \,\rangle \longrightarrow_{\beta,\mu} \langle\, u_1, \ldots, u'_i, \ldots, u_p \,\rangle = t'^p$.

By induction hypothesis $[\![\, u_i \,]\!] \longrightarrow^*_{\beta,\mu} [\![\, u'_i \,]\!]$. Therefore:

$$\begin{aligned}[\![\, t^p \,]\!] &= \langle\, [\![\, u_1 \,]\!], \ldots, [\![\, u_i \,]\!], \ldots, [\![\, u_p \,]\!] \,\rangle \\ &\longrightarrow^*_{\beta,\mu} \langle\, [\![\, u_1 \,]\!], \ldots, [\![\, u'_i \,]\!], \ldots, [\![\, u_p \,]\!] \,\rangle \\ &= [\![\, t'^p \,]\!]\end{aligned}$$

- If $p \neq 0$ and

$$[\phi] = [\vec{\chi_p} \leftarrow \tau] \longrightarrow_{\beta,\mu} [\vec{\chi_p} \leftarrow \tau'] = [\phi']$$

  By induction hypothesis $[\![\tau]\!] \longrightarrow^*_{\beta,\mu} [\![\tau']\!]$. Therefore:

$$\{\!| \phi |\!\} = \{\!| [\![\tau]\!]/\chi_p |\!\} \longrightarrow^*_{\beta,\mu} \{[\![\tau']\!]/\chi_p\} = \{\!| \phi' |\!\}$$

- If $[\phi] = [\leftarrow \tau] \longrightarrow_{\beta,\mu} [\leftarrow \tau'] = [\phi']$. Then $\{\!| \phi |\!\} = \emptyset = \{\!| \phi' |\!\}$.

- If $p \neq 0$ and

$$[\phi] = [\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{u_p} \,\rangle[\Psi]] \longrightarrow_{\beta,\mu} [\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{u'_p} \,\rangle[\Psi']] = [\phi']$$

  By induction hypothesis

$$[\![\langle\, \vec{u_p} \,\rangle[\Psi]]\!] \longrightarrow^*_{\beta,\mu} [\![\langle\, \vec{u'_p} \,\rangle[\Psi']]\!]$$

$$\begin{aligned} \{\!| \phi |\!\} &= \{\mathcal{A}y.[\![u^i[\Psi]]\!]/x_i\} \\ &\longrightarrow^*_{\beta,\mu} \{\mathcal{A}y.[\![u'^i[\Psi']]\!]/x_i\} \\ &= \{\!| \phi' |\!\} \end{aligned}$$

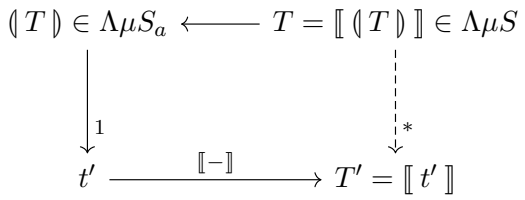
- If $[\phi] = [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]] \longrightarrow_{\beta,\mu} [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi']] = [\phi']$

$$\{\!| \phi |\!\} = \emptyset = \{\!| \phi' |\!\}$$

□

By Theorem 3.0.3, we know that one β, μ step in the atomic calculus gives zero or more β, μ steps in the $\Lambda\mu S$-calculus, showing that the atomic calculus is well-behaved. The following theorem states that after applying atomic reductions on the translation $(\!| T |\!)$ of a $\Lambda\mu S$-term T, interpreting the result back in the $\Lambda\mu S$-calculus corresponds to β/μ-reduction. For a $\Lambda\mu S$-term T, if its atomic translation $(\!| T |\!)$ reduces to t', then T reduces in zero or more steps to $[\![t']\!]$:

Theorem 3.0.4 (Atomic reduction implements $\Lambda\mu S$-reduction). For any $T \in \Lambda\mu S$, if $(\!| T |\!) \longrightarrow^1 t'$ then $[\![(\!| T |\!)]\!] = T \longrightarrow^* T'$ for a certain $T' = [\![t']\!] \in \Lambda\mu S$.



Proof. Suppose T is such that $(\!|\, T \,|\!) \longrightarrow^1 t'$. Let $T' = [\![\, t' \,]\!]$.

By Lemma 3.0.1 $T = [\![\, (\!|\, T \,|\!) \,]\!]$, and:

- If $(\!|\, T \,|\!) \longrightarrow_s t'$, by Lemma 3.0.2 $T = [\![\, (\!|\, T \,|\!) \,]\!] = [\![\, t' \,]\!] = T'$.
- If $(\!|\, T \,|\!) \longrightarrow_{\beta,\mu} t'$, by Theorem 3.0.3 $T = [\![\, (\!|\, T \,|\!) \,]\!] \longrightarrow^*_{\beta,\mu} [\![\, t' \,]\!] = T'$.

□

Sharing reductions are strongly normalizing and $[\![–]\!]$ takes a path in the atomic calculus to a corresponding path the in $\Lambda\mu S$-calculus:

$$(\!|\, T \,|\!) = t \longrightarrow t_1 \longrightarrow t_2 \longrightarrow \dots$$
$$[\![\, t \,]\!] = T \longrightarrow^* [\![\, t_1 \,]\!] \longrightarrow^* [\![\, t_2 \,]\!] \dots$$

The remaining task to show PSN is to take an infinite path in the atomic calculus to an infinite path in the $\Lambda\mu S$-calculus. Since sharing reductions are strongly normalizing, we focus on (β, μ) reductions. Reductions inside weakenings are the only obstacle left, take the following example:

Example 3.0.5. Let $T = (\lambda y.x)\,\Omega$, where Ω is the usual term with an infinite reduction. Then $T \longrightarrow_\beta T' = x$, and

$$(\!|\, T \,|\!) = (\lambda y.x[\leftarrow y])\,(\!|\, \Omega \,|\!) \longrightarrow_\beta t' = x[\leftarrow (\!|\, \Omega \,|\!)]$$

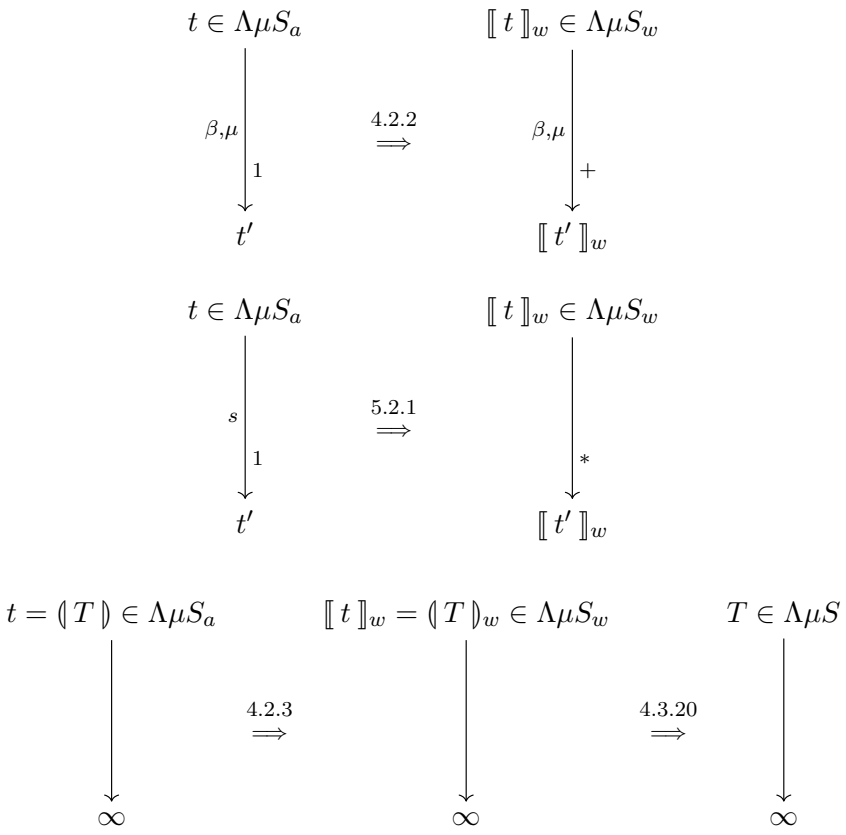
The term $[\![\, t' \,]\!] = T' = x$ is in normal form, however there exists an infinite reduction path from t'.

We thus need to work with reduction strategies that would preserve infinite paths, by preventing diverging terms from falling into weakenings. Chapter 4 provides a strategy preventing this situation from happening. From there, we finally obtain PSN.

Chapter 4

The weakening calculus

For our proof of PSN, we would like to construct an infinite reduction path in the $\Lambda\mu S$-calculus from an infinite reduction path in the atomic $\Lambda\mu S_a$-calculus. However, some β, μ infinite reductions may fall inside weakenings, making one atomic reduction step correspond to zero $\Lambda\mu S$-steps. We therefore cannot directly construct an infinite path in the $\Lambda\mu S$-calculus from an infinite path in the atomic calculus. For these purposes, we use an intermediate calculus, the weakening calculus $\Lambda\mu S_w$, between the $\Lambda\mu S$-calculus and the atomic $\Lambda\mu S_a$-calculus. Our proof for PSN is then split into two parts. First we get a straightforward proof for PSN between the atomic calculus and the weakening calculus (i.e. $[\![-]\!]_w$ preserves non-termination, Theorem 4.2.3), by translating an infinite reduction in the atomic calculus to an infinite reduction in the weakening calculus. In particular, we show that a β, μ-step in the atomic calculus gives at least one step in the weakening calculus ($[\![-]\!]_w$ strictly preserves β, μ-reductions, Lemma 4.2.2), and that a sharing atomic step gives zero or more steps in the weakening calculus ($[\![-]\!]_w$ commutes with sharing reduction, Theorem 5.2.1). The remaining part, PSN between the weakening calculus and the $\Lambda\mu S$-calculus, isolates the hard part of the proof, and requires looking at strategies. This part is shown in this chapter. By combining these two intermediate PSN results, and using the strong normalization of sharing reductions (shown in Chapter 5), we obtain PSN for the atomic $\lambda\mu$-calculus with respect to the $\lambda\mu$-calculus.



4.1 The weakening calculus $\Lambda\mu S_w$

In this section we define the weakening calculus $\Lambda\mu S_w$ and its reduction rules. The calculus unfolds sharing (interpreted from the atomic calculus by substitutions i.e. duplications) while keeping weakenings. Instead of using a distributor as in $\Lambda\mu S_a$ that would only be used for abstractions, we add a new term $\bullet$ which corresponds to a bound variable x in a distributor $[\twoheadleftarrow \mathcal{A}x.t]$.

4.1.1 Syntax and translations $(\!|-|\!)_w, [\![-]\!]_w$

In the weakening calculus, variables do not occur linearly as in the atomic calculus $\Lambda\mu S_a$, however, bound variables can occur zero times or more. The syntax of the

weakening calculus $\Lambda\mu S_w$ is as follows:

$$\begin{array}{l} \textit{Terms } T, U ::= \ x \mid \lambda x.T^{\,(*)} \mid (T)U \mid (T)\mathcal{S} \mid \mu\alpha.T^{\,(*)} \mid U[\phi] \mid \bullet^{(**)} \\ \textit{Streams } \mathcal{S} ::= \ \alpha \mid T \circ \mathcal{S} \mid \mathcal{S}[\phi] \\ \textit{Closures } [\phi], [\psi] ::= \ [\leftarrow U] \mid [\leftarrow \mathcal{S}] \end{array}$$

$^{(*)}$ where $x, \alpha \in FV(T)$ and $^{(**)}$ the bullet $\bullet$ may only occur inside weakenings.

Notation 2. We denote:

1. terms and streams by U^* (and we call U^* an *expression* of $\Lambda\mu S_w$),
2. applications $(T)U^*, T \circ U^*$ by $@(T, U^*)$,
3. abstractions $\lambda x.T, \mu\alpha.T\{\alpha/x\}$ by $\mathcal{A}x.T$.

We can now define the translation $(\!|-|\!)_w$ of $\Lambda\mu S$-terms to weakening terms:

Definition 4.1.1. [Translation $\Lambda\mu S \xrightarrow{(\!|-|\!)_w} \Lambda\mu S_w$] Let χ be a variable.

- $(\!|\, \chi \,|\!)_w = \chi$
- $(\!|\, @(t, u^*) \,|\!)_w = @((\!|\, t \,|\!), (\!|\, u^* \,|\!))$
- $(\!|\, \mathcal{A}x.t \,|\!)_w = \begin{cases} \mathcal{A}x.(\!|\, t \,|\!)_w & \text{if } x \in FV(t) \\ \mathcal{A}x.((\!|\, t \,|\!)_w[\leftarrow x]) & \text{otherwise} \end{cases}$

We now define the translation $[\![-]\!]_w$ from atomic terms to weakening terms, as well as the auxiliary function $\{\!|-|\!\}_w$ translating closures as substitutions or weakenings:

Definition 4.1.2. [Translation $\Lambda\mu S_a \xrightarrow{[\![-]\!]_w} \Lambda\mu S_w$] Let $\tau \in \mathbb{T} \cup \mathbb{S}$.

1. $[\![\, x \,]\!]_w = x$
2. $[\![\, \mathcal{A}x.t \,]\!]_w = \mathcal{A}x.[\![\, t \,]\!]_w$
3. $[\![\, @(t, \tau) \,]\!]_w = @([\![\, t \,]\!]_w, [\![\, \tau \,]\!]_w)$
4. $[\![\, u^*[\phi] \,]\!]_w = [\![\, u^* \,]\!]_w \{\!|\, \phi \,|\!\}_w$
5. $\{\!|\, \vec{\chi_p} \leftarrow \tau \,|\!\}_w = \begin{cases} [\leftarrow [\![\, \tau \,]\!]_w] & \text{if } p = 0 \\ \{[\![\, \tau \,]\!]_w / \chi_i\}_{1 \le i \le p} & \text{if } p \ge 1 \end{cases}$

6. $\{\!|\, \vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p}\,\rangle[\Psi]\,|\!\}_w = \begin{cases} \{\!|\, \Psi\,|\!\}_w\{\bullet/y\} & \text{if } p = 0 \\ \{\mathcal{A}y.[\![\, t_i[\Psi]\,]\!]_w/x_i\}_{1\leq i\leq p} & \text{if } p \geq 1 \end{cases}$
7. $\{\!|\, \Psi\,|\!\} = \{\!|\, \psi_1\,|\!\} \ldots \{\!|\, \psi_n\,|\!\}$ where $[\Psi] = [\psi_1] \ldots [\psi_n]$

Another translation $\lfloor - \rfloor$ from weakening terms to $\Lambda\mu S$-terms is done, discarding all weakenings. The $\bullet$ symbol appears in weakenings of $\Lambda\mu S_w$, and therefore is discarded with $\lfloor - \rfloor$:

Definition 4.1.3. [Translation $\Lambda\mu S_w \xrightarrow{\lfloor - \rfloor} \Lambda\mu S$] Let χ be a variable, let τ be a term or a stream. Then:

1. $\lfloor \chi \rfloor = \chi$
2. $\lfloor \mathcal{A}x.T \rfloor = \mathcal{A}x.\lfloor T \rfloor$
3. $\lfloor @(T, U^*) \rfloor = @(\lfloor T \rfloor, \lfloor U^* \rfloor)$
4. $\lfloor U[\leftarrow T^*] \rfloor = \lfloor U \rfloor$

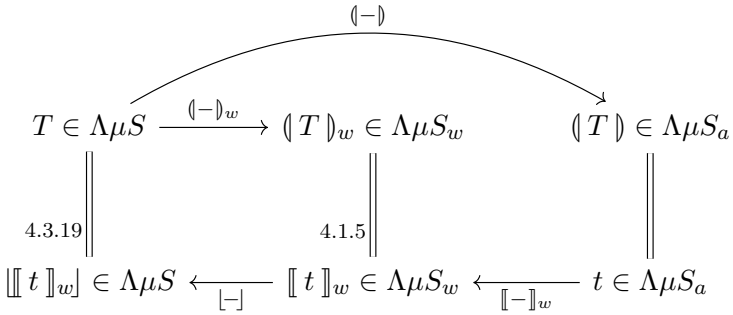
We need to define subexpressions of weakenings terms to describe reduction rules:

Definition 4.1.4. [Subexpressions in $\Lambda\mu S_w$] Let $U^*, V^* \in \Lambda\mu S_w$. We define by induction on U^* the set $\mathcal{E}(U^*)$ of subexpressions of U^*:

1. $\mathcal{E}(x) = \{x\}$
2. $\mathcal{E}(\mathcal{A}x.T) = \{\mathcal{A}x.T\} \cup \mathcal{E}(T)$
3. $\mathcal{E}(@(T, V^*)) = \{@(T, V^*)\} \cup \mathcal{E}(T) \cup \mathcal{E}(V^*)$
4. $\mathcal{E}(V^*[\phi]) = \{V^*[\phi]\} \cup \mathcal{E}(V^*) \cup \mathcal{E}(\phi)$
5. $\mathcal{E}(V^*[\leftarrow U^*]) = \{V^*[\leftarrow U^*]\} \cup \mathcal{E}(V^*) \cup \mathcal{E}(U^*)$
6. $\mathcal{E}(\bullet) = \emptyset$

We can observe that for a $\Lambda\mu S$-term T, its weakening translation $(\!|\, T\,|\!)_w$ corresponds to the denotation in the weakening calculus of its atomic translation $(\!|\, T\,|\!)$, making the

weakening calculus act as a bridge between $\Lambda\mu S$-terms and their atomic translation. We will show the following:



In particular, we have two retractions from $\Lambda\mu S_a$ to $\Lambda\mu S$, and from $\Lambda\mu S_w$ to $\Lambda\mu S$. However, we do not have a retraction from $\Lambda\mu S_a$ to $\Lambda\mu S_w$. For example, a term in the weakening calculus such as $T_w = x[\leftarrow (\bullet)\bullet]$ could come from

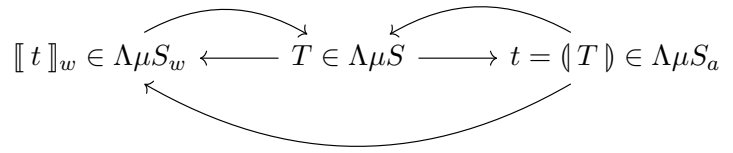
$$x[\twoheadleftarrow \lambda y.\langle\ \rangle[\twoheadleftarrow \lambda z.\langle\ \rangle[\leftarrow (y)z]]]$$

(after reducing from $x[\leftarrow \lambda y.\lambda z.(y)z]$)

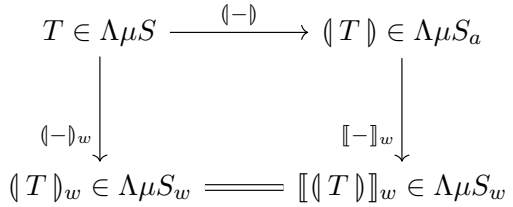
but also from

$$x[\twoheadleftarrow \lambda y.\langle\ \rangle[\leftarrow (y_1)y_2[y_1, y_2 \leftarrow y]]]$$

(after reducing from $x[\leftarrow \lambda y.(y_1)y_2[y_1, y_2 \leftarrow y]]$).



Lemma 4.1.5. For $T \in \Lambda\mu S$, $[\![\, (\!|\, T \,|\!) \,]\!]_w = (\!|\, T \,|\!)_w$.



Proof. The proof is similar to Lemma 3.0.1 (Interpretation is inverse to translation).

□

4.1.2 Weakening reductions

From the atomic calculus to the weakening calculus, weakenings are kept, while other sharings are interpreted with duplications (substitutions). The reductions rules in the weakening calculus aim to capture atomic rules for nullary sharings and distributors.

Weakening reductions are presented below, and will be explained after:

Definition 4.1.6. [Weakening removals in $\Lambda\mu S_w$]

Lifting rules

1. $\mathcal{A}x.(U[\phi]) \longrightarrow_w (\mathcal{A}x.U)[\phi]$ if $x \in FV(U)$
2. $\begin{matrix} @(U[\phi], T^*) \\ @(U, T^*[\phi]) \end{matrix} \longrightarrow_w @(U, T^*)[\phi]$
3. $U^*[\leftarrow T^*[\phi]] \longrightarrow_w U^*[\leftarrow T^*][\phi]$

Compounding rules

1. $U^*[\leftarrow T^*] \longrightarrow_w U^*$ if T^* is a subexpression of U^*

Duplication rules

1. $U^*[\leftarrow @(V, T^*)] \longrightarrow_w U^*[\leftarrow V][\leftarrow T^*]$
2. $U^*[\leftarrow \mathcal{A}x.T] \longrightarrow_w U^*[\leftarrow T\{\bullet/x\}]$
3. $U^*[\leftarrow \bullet] \longrightarrow_w U^*$

The $\bullet$ captures the deleted variable in a distributor. In the atomic calculus, a nullary distributor $[\twoheadleftarrow \mathcal{A}x.\langle\rangle[\leftarrow x]]$ could be translated in the weakening calculus by a weakened variable $[\leftarrow x]$, but this would create α-equivalence issues as

$$[\twoheadleftarrow \mathcal{A}x.\langle\ \rangle[\leftarrow x]] =_\alpha [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\leftarrow y]]$$

when $[\leftarrow x] \neq [\leftarrow y]$. For this reason we use $\bullet$ instead of a variable.

Example 4.1.7. In the atomic calculus:

$$\begin{aligned}[\leftarrow \lambda x.(x)(y)z] &\longrightarrow_s [\twoheadleftarrow \lambda x.\langle\ \rangle[\leftarrow (x)(y)z]] \\ &\longrightarrow_s [\twoheadleftarrow \lambda x.\langle\ \rangle[\leftarrow x][\leftarrow (y)z]] \\ &\longrightarrow_s [\twoheadleftarrow \lambda x.\langle\ \rangle[\leftarrow x]][\leftarrow (y)z] \\ &\longrightarrow_s [\leftarrow (y)z]\end{aligned}$$

In the weakening calculus:

$$\begin{aligned}[\leftarrow \lambda x.(x)(y)z] &\longrightarrow_w [\leftarrow ((x)(y)z)\{\bullet/x\}] \\ &= [\leftarrow (\bullet)(y)z] \\ &\longrightarrow_w [\leftarrow \bullet][\leftarrow (y)z] \\ &\longrightarrow_w [\leftarrow (y)z]\end{aligned}$$

Now consider $U^*[\leftarrow T^*]$ where T^* is a subexpression of U^*. With the other rules in the weakening calculus, this term can eventually reduce to $U^*[\leftarrow \chi_1]\ldots[\leftarrow \chi_p]$ where $\chi_1, \ldots, \chi_p = FV(T^*)$. Consider the following example:

Example 4.1.8. In the atomic calculus:

$$(x)y_1[\leftarrow y_2][y_1, y_2 \leftarrow y] \longrightarrow_s (x)y_1[y_1 \leftarrow y] \longrightarrow_s (x)y$$

The weakening $[\leftarrow y_2]$ is removed by applying a compounding rule. In the weakening calculus the term is translated to $(x)y[\leftarrow y]$, and we would like to have a rule to remove the weakening $[\leftarrow y]$. A rule in the weakening calculus that would correspond to the compounding rule in the atomic calculus would be:

$$U^*[\leftarrow \chi] \longrightarrow_w U^*$$

if $\chi \in FV(U^*)$. We would then have:

$$(x)y[\leftarrow y] \longrightarrow_s (x)y$$

We could therefore add the rule $U^*[\leftarrow \chi] \longrightarrow_w U^*$ if $\chi \in FV(U^*)$, but instead we use a more general rule:

$$U^*[\leftarrow T^*] \longrightarrow_w U^*$$

if T^* is a subexpression of U^*.

In the example below, we see that applying the general rule saves several reduction steps, and therefore is more convenient to use.

Example 4.1.9. Let $t = \lambda a.(a)z = [\![\, t \,]\!]_w = T$. In the atomic calculus:

$$\begin{aligned}(x)(\lambda k.(k)z_1)u[\leftarrow \lambda k.(k)z_2][z_1, z_2 \leftarrow t] &\longrightarrow_s^* (x)(\lambda k.(k)z_1)u[\leftarrow z_2][z_1, z_2 \leftarrow t] \\ &\longrightarrow_s^* (x)(\lambda k.(k)t)u \\ &= (x)(\lambda k.(k)(\lambda a.(a)z))u\end{aligned}$$

In the weakening calculus:

$$\begin{aligned}[\![\, (x)(\lambda k.(k)z_1)u[\leftarrow \lambda k.(k)z_2][z_1, z_2 \leftarrow t] \,]\!]_w &= (x)(\lambda k.(k)T)u[\leftarrow \lambda k.(k)T] \\ &\longrightarrow_w (x)(\lambda k.(k)T)u \\ &= (x)(\lambda k.(k)(\lambda a.(a)z))u\end{aligned}$$

This rule can also come from lifting sharings:

Example 4.1.10. Let $t, u \in \Lambda\mu S_a$, let $T = [\![\, t \,]\!]_w$, and $U = [\![\, u \,]\!]_w$. In the atomic calculus:

$$(x_1)x_2[x_1, x_2 \leftarrow t[\leftarrow u]] \longrightarrow_s (x_1)x_2[x_1, x_2 \leftarrow t][\leftarrow u]$$

In the weakening calculus:

$$\begin{aligned}&[\![\, (x_1)x_2[x_1, x_2 \leftarrow t[\leftarrow u]] \,]\!]_w \\ &= (T[\leftarrow U])(T[\leftarrow U]) \\ &\longrightarrow_w^* (((T)T)[\leftarrow U])[\leftarrow U] \\ &\longrightarrow_w (T)T[\leftarrow U]\end{aligned}$$

4.2 Properties of $[\![-]\!]_w$

In this section we show that $[\![-]\!]_w$ strictly preserves β, μ-reductions. Recall that the weakening calculus has explicit weakenings $[\leftarrow U]$, but no sharings $[x_1, \ldots, x_p \leftarrow U]$ where $p > 0$. For distributors, we introduced a new $\bullet$ symbol, which translates in the weakening calculus a bound variable x (respectively α) in a nullary distributor $[\twoheadleftarrow \lambda x.\langle\ \rangle[\Phi]]$ (respectively $[\twoheadleftarrow \mu\alpha.\langle\ \rangle[\Phi]]$) from the atomic calculus. This bound variable could be represented in the weakening calculus by a free variable, however we do not want this variable to become accidentally bound later, therefore we use $\bullet$ to represent it instead. For a $\Lambda\mu S$-term T, $(\!|\, T \,|\!)_w$ is its translation in the weakening

calculus. For an atomic term t, $[\![\, t \,]\!]_w$ is its translation in the weakening calculus. For a weakening term U, the term $\lfloor U \rfloor \in \Lambda\mu S$ corresponds to U where all the weakenings have been removed.

We will combine two theorems:

- $[\![-]\!]_w$ preserves non-termination 4.2.3
- PSN for the weakening calculus 4.3.20

$$\begin{array}{ccc} t \in \Lambda\mu S_a & & [\![\, t \,]\!]_w \in \Lambda\mu S_w \\ \Big\downarrow{\scriptstyle \beta,\mu}\,{\scriptstyle 1} & \overset{4.2.2}{\Longrightarrow} & \Big\downarrow{\scriptstyle \beta,\mu}\,{\scriptstyle +} \\ t' & & [\![\, t' \,]\!]_w \end{array}$$

$$\begin{array}{ccc} t \in \Lambda\mu S_a & \xrightarrow{[\![-]\!]_w} & [\![\, t \,]\!]_w \in \Lambda\mu S_w \\ \Big\downarrow{\scriptstyle s} & & \Big\downarrow{\scriptstyle *} \\ u \in \Lambda\mu S_a & \xrightarrow{[\![-]\!]_w} & [\![\, u \,]\!]_w \in \Lambda\mu S_w \end{array}$$

First, Theorem 4.2.3 is true by design of the weakening calculus (and because of strong normalization of sharing reductions), one (β, μ) step in the atomic calculus corresponds to at least one step in the weakening calculus (Lemma 4.2.2). The second theorem is less direct to show, and requires using a strategy that would find a *perpetual* reduction path in the weakening calculus corresponding to an infinite path in the $\Lambda\mu S$-calculus. The core idea is that if an infinite reduction of $(\!|\, T \,|\!)_w$ was occurring inside a weakening, it could have occurred outside the weakening as well.

The lemma below shows that, from a $\Lambda\mu S$-term T, one reduction step leads to at least a (β, μ)-reduction step in the weakening calculus.

Lemma 4.2.1. Let $T \in \Lambda\mu S$. If $(\!|\, T \,|\!) \longrightarrow_{\beta,\mu} t'$ then $[\![\, (\!|\, T \,|\!) \,]\!]_w \longrightarrow^{+}_{\beta,\mu} [\![\, t' \,]\!]_w$.

$$\begin{array}{ccc} (\!|\, T \,|\!) \in \Lambda\mu S_a & & [\![\, (\!|\, T \,|\!) \,]\!]_w = (\!|\, T \,|\!)_w \in \Lambda\mu S_w \\ \Big\downarrow{\scriptstyle \beta,\mu} & \overset{4.2.1}{\Longrightarrow} & \Big\downarrow{\scriptstyle +}\,{\scriptstyle \beta,\mu} \\ t' & & [\![\, t' \,]\!]_w \end{array}$$

Proof. By induction on T. Let τ be a term or a stream, let χ, χ_i be variables. Recall that $\frac{l_i}{\chi_i}$ means replacing χ_i with l_i fresh distinct variables. Recall that

$$\begin{aligned}(\!| T |\!) =& (\!| T \frac{l_1}{x_1} \dots \frac{l_p}{x_p} \frac{l_{p+1}}{\alpha_1} \dots \frac{l_{p+k}}{\alpha_k} |\!)'[(\vec{x_1})_{l_1} \leftarrow x_1] \dots [(\vec{x_p})_{l_p} \leftarrow x_p] \\ & [(\vec{\alpha_1})_{l_{p+1}} \leftarrow \alpha_1] \dots [(\vec{\alpha_k})_{l_{p+k}} \leftarrow \alpha_k]\end{aligned}$$

Let $\sigma = \underbrace{\frac{l_1}{\chi_1} \dots \frac{l_p}{\chi_p}}_{\text{occurrences in } \tau}$, replacing the different occurrences of each free variable χ_i with fresh, distinct variables. Let $\Phi = [\vec{\chi_1}^{l_1} \leftarrow \chi_1] \dots [\vec{\chi_p}^{l_p} \leftarrow \chi_p]$. We use the inductive hypothesis: $[\![(\!| \tau\sigma |\!)']\!]_w = (\!| \tau |\!)_w\sigma$. Also, since $\{\!| \Phi |\!\}_w$ and σ are inverse operations, we have

$$[\![(\!| T |\!)]\!]_w = [\![(\!| T\sigma |\!)'\Phi]\!]_w = [\![(\!| T\sigma |\!)']\!]_w \{\!| \Phi |\!\}_w = (\!| T |\!)_w\sigma\{\!| \Phi |\!\}_w = (\!| T |\!)_w$$

- If $\tau = \chi$, then $[\![(\!| \chi |\!)']\!]_w = [\![\chi]\!]_w = \chi = (\!| \chi |\!)_w$.
- Let $T = @(U, \tau)$. Then $\sigma = \sigma_1\sigma_2 = \underbrace{\frac{l_1}{\chi_1} \dots \frac{l_k}{\chi_k}}_{\text{occurrences in } U} \underbrace{\frac{l_{k+1}}{\chi_{k+1}} \dots \frac{l_p}{\chi_p}}_{\text{occurrences in } \tau}$.
  - If $T = @(\mathcal{A}x.U, \tau)$:
    1. Suppose $|u|_x = p \neq 1$.

$$\begin{aligned}(\!| (\lambda x.u)v |\!) &= (\!| ((\lambda x.u)v)\sigma |\!)'\Phi \\ &= ((\lambda x.(\!| u\sigma_1 |\!)')(\!| v\sigma_2 |\!)')\Phi \\ &\longrightarrow_{\beta_t} ((\!| u\sigma_1 |\!)'[x \leftarrow (\!| v\sigma_2 |\!)'])\Phi\end{aligned}$$

Then:

$$\begin{aligned}[\![(\!| (\lambda x.u)v |\!)]\!]_w &= [\![(\!| (\lambda x.u)v |\!)'\Phi]\!]_w \\ &= [\![((\lambda x.(\!| u\sigma_1 |\!)')(\!| v\sigma_2 |\!)')\Phi]\!]_w \\ &= ((\lambda x.[\![(\!| u\sigma_1 |\!)']\!]_w)[\![(\!| v\sigma_2 |\!)']\!]_w)\{\!| \Phi |\!\}_w \\ &= ((\lambda x.(\!| u\sigma_1 |\!)_w)(\!| v\sigma_2 |\!)_w)\{\!| \Phi |\!\}_w \\ &\longrightarrow_{\beta_t} (\!| u |\!)_w\{(\!| v |\!)_w/x\}\sigma_1\sigma_2\{\!| \Phi |\!\}_w \\ &= (\!| u |\!)_w\{(\!| v |\!)_w/x\}\sigma\{\!| \Phi |\!\}_w \\ &= [\![((\!| u\sigma_1 |\!)'\{(\!| v\sigma_2' |\!)/x\})\Phi]\!]_w\end{aligned}$$

2. Same for other cases.

– Otherwise, if $(\!|\, @(U,\tau) \,|\!) = @((\!|\, U \,|\!),(\!|\, \tau \,|\!)) \longrightarrow_{\beta,\mu} t'$, then either $(\!|\, U \,|\!) \longrightarrow_{\beta,\mu} u'$ or $(\!|\, \tau \,|\!) \longrightarrow_{\beta,\mu} \tau'$. Suppose $(\!|\, U \,|\!) \longrightarrow_{\beta,\mu} u'$.

By induction hypothesis:

$$\begin{aligned}
[\![\, (\!|\, U \,|\!) \,]\!]_w &= [\![\, (\!|\, U\sigma_1 \,|\!)'\Phi \,]\!]_w \\
&= (\!|\, U \,|\!)_w\sigma_1\{\!|\, \Phi \,|\!\}_w \\
&\longrightarrow^+_{\beta,\mu} [\![\, u' \,]\!]_w
\end{aligned}$$

Therefore,

$$\begin{aligned}
[\![\, (\!|\, T \,|\!) \,]\!]_w &= @([\![\, (\!|\, U\sigma_1 \,|\!)' \,]\!]_w, [\![\, (\!|\, \tau\sigma_2 \,|\!)' \,]\!]_w)\{\!|\, \Phi \,|\!\}_w \\
&= @([\![\, (\!|\, U\sigma_1 \,|\!)' \,]\!]_w\{\!|\, \Phi \,|\!\}_w, [\![\, (\!|\, \tau\sigma_2 \,|\!)' \,]\!]_w\{\!|\, \Phi \,|\!\}_w) \\
&= @((\!|\, U \,|\!)_w\sigma_1\{\!|\, \Phi \,|\!\}_w, [\![\, (\!|\, \tau\sigma_2 \,|\!)' \,]\!]_w\{\!|\, \Phi \,|\!\}_w) \\
&\longrightarrow^+_{\beta,\mu} @([\![\, u' \,]\!]_w, [\![\, (\!|\, \tau \,|\!) \,]\!]_w)
\end{aligned}$$

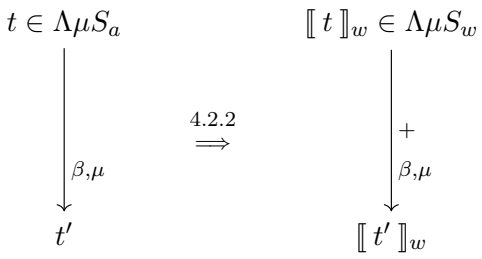
- Let $T = \mathcal{A}x.U$. If $(\!|\, T \,|\!) \longrightarrow_{\beta,\mu} t'$, then $(\!|\, U \,|\!) \longrightarrow_{\beta,\mu} u'$, where $t' = \mathcal{A}x.u'$. Then:

$$\begin{aligned}
[\![\, (\!|\, \mathcal{A}x.U \,|\!) \,]\!]_w &= [\![\, \mathcal{A}x.(\!|\, U \,|\!) \,]\!]_w \\
&= \mathcal{A}x.[\![\, (\!|\, U \,|\!) \,]\!]_w \\
&\longrightarrow^+_{\beta,\mu} \mathcal{A}x.[\![\, u' \,]\!]_w \\
&= [\![\, \mathcal{A}x.u' \,]\!]_w
\end{aligned}$$

□

The lemma below is the main result that shows Theorem 4.2.3, stating that a (β,μ)-reduction in the atomic calculus corresponds to at least a (β,μ)-reduction in the weakening calculus.

Lemma 4.2.2 ($[\![-]\!]_w$ strictly preserves β,μ-reductions). Let $t \in \Lambda\mu S_a$. If $t \longrightarrow_{\beta,\mu} t'$ then $[\![\, t \,]\!]_w \longrightarrow^+_{\beta,\mu} [\![\, t' \,]\!]_w$.



Proof. By induction on t.

- If t is a redex:
  - If $t = (\lambda x.u)v \longrightarrow_\beta u\{v/x\} = t'$. Then

$$\begin{aligned}[\![t]\!]_w &= (\lambda x.[\![u]\!]_w)[\![v]\!]_w \\ &\longrightarrow_\beta [\![u]\!]_w\{[\![v]\!]_w/x\} \\ &= [\![t']\!]_w\end{aligned}$$

  - The other cases are similar.
- If $t = @(u,\tau) \longrightarrow_{\beta,\mu} @(u',\tau) = t'$. By induction hypothesis,

$$[\![u]\!]_w \longrightarrow^+_{\beta,\mu} [\![u']\!]_w$$

  Therefore

$$[\![t]\!]_w = @([\![u]\!]_w, [\![\tau]\!]_w) \longrightarrow^+_{\beta,\mu} @([\![u']\!]_w, [\![\tau]\!]_w) = [\![t']\!]_w$$

  The proof is similar if $t = @(u,\tau) \longrightarrow_{\beta,\mu} @(u,\tau') = t'$.
- If $t = \mathcal{A}x.u \longrightarrow_{\beta,\mu} \mathcal{A}x.u' = t'$. By induction hypothesis,

$$[\![u]\!]_w \longrightarrow^+_{\beta,\mu} [\![u']\!]_w$$

  Therefore

$$[\![t]\!]_w = \mathcal{A}x.[\![u]\!]_w \longrightarrow^+_{\beta,\mu} \mathcal{A}x.[\![u']\!]_w = [\![t']\!]_w$$

- If $t^* = u^*[\phi] \longrightarrow_{\beta,\mu} u^{*\prime}[\phi] = t^{*\prime}$. By induction hypothesis,

$$[\![u^*]\!]_w \longrightarrow^+_{\beta,\mu} [\![u^{*\prime}]\!]_w$$

Therefore

$$\begin{aligned}[\![\, t^* \,]\!]_w &= [\![\, u^* \,]\!]_w \{\!|\, \phi \,|\!\}_w \\ &\longrightarrow^+_{\beta,\mu} [\![\, u^{*\prime} \,]\!]_w \{\!|\, \phi \,|\!\}_w \\ &= [\![\, t^{*\prime} \,]\!]_w\end{aligned}$$

- If $t^p = \langle\, u_1, \ldots, u_i, \ldots, u_p \,\rangle \longrightarrow_{\beta,\mu} \langle\, u_1, \ldots, u'_i, \ldots, u_p \,\rangle = t'^p$.

  By induction hypothesis

  $$[\![\, u_i \,]\!]_w \longrightarrow^+_{\beta,\mu} [\![\, u'_i \,]\!]_w$$

  Therefore

  $$\begin{aligned}[\![\, t^p \,]\!]_w &= \langle\, [\![\, u_1 \,]\!]_w, \ldots, [\![\, u_i \,]\!]_w, \ldots, [\![\, u_p \,]\!]_w \,\rangle \\ &\longrightarrow^+_{\beta,\mu} \langle\, [\![\, u_1 \,]\!]_w, \ldots, [\![\, u'_i \,]\!]_w, \ldots, [\![\, u_p \,]\!]_w \,\rangle \\ &= [\![\, t'^p \,]\!]_w\end{aligned}$$

- If $p \neq 0$ and

  $$[\phi] = [\vec{\chi_p} \leftarrow \tau] \longrightarrow_{\beta,\mu} [\vec{\chi_p} \leftarrow \tau'] = [\phi']$$

  By induction hypothesis

  $$[\![\, \tau \,]\!]_w \longrightarrow^+_{\beta,\mu} [\![\, \tau' \,]\!]_w$$

  Therefore

  $$\begin{aligned}\{\!|\, \phi \,|\!\}_w &= \{\!|\, [\![\, \tau \,]\!]_w / \chi_p \,|\!\}_w \\ &\longrightarrow^+_{\beta,\mu} \{[\![\, \tau' \,]\!]_w / \chi_p\} \\ &= \{\!|\, \phi' \,|\!\}_w\end{aligned}$$

- If $[\phi] = [\leftarrow \tau] \longrightarrow_{\beta,\mu} [\leftarrow \tau'] = \phi'$. By induction hypothesis

  $$[\![\, \tau \,]\!]_w \longrightarrow^+_{\beta,\mu} [\![\, \tau' \,]\!]_w$$

  Therefore

  $$\begin{aligned}\{\!|\, \phi \,|\!\}_w &= (\leftarrow [\![\, \tau \,]\!]_w) \\ &\longrightarrow^+_{\beta,\mu} (\leftarrow [\![\, \tau' \,]\!]_w) \\ &= \{\!|\, \phi' \,|\!\}_w\end{aligned}$$

- If $p \neq 0$ and

$$[\phi] = [\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{u_p} \,\rangle[\Psi]] \longrightarrow_{\beta,\mu} [\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{u'_p} \,\rangle[\Psi']] = [\phi']$$

  By induction hypothesis

$$[\![\, \langle\, \vec{u_p} \,\rangle[\Psi] \,]\!]_w \longrightarrow^+_{\beta,\mu} [\![\, \langle\, \vec{u'_p} \,\rangle[\Psi'] \,]\!]_w$$

$$\begin{aligned} \{\!|\, \phi \,|\!\}_w &= \{\mathcal{A}y.[\![\, u^i[\Psi] \,]\!]_w / x_i\} \\ &\longrightarrow^+_{\beta,\mu} \{\mathcal{A}y.[\![\, u'^i[\Psi'] \,]\!]_w / x_i\} \\ &= \{\!|\, \phi' \,|\!\}_w \end{aligned}$$

- If $[\phi] = [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]] \longrightarrow_{\beta,\mu} [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi']] = [\phi']$

  By induction hypothesis

$$[\![\, \langle\ \rangle[\Psi] \,]\!]_w \longrightarrow^+_{\beta,\mu} [\![\, \langle\ \rangle[\Psi'] \,]\!]_w$$

$$\begin{aligned} \{\!|\, \phi \,|\!\}_w &= \{\bullet/y\}\{\!|\, \Phi \,|\!\}_w \\ &\longrightarrow^+_{\beta,\mu} \{\bullet/y\}\{\!|\, \Phi' \,|\!\}_w \\ &= \{\!|\, \phi' \,|\!\}_w \end{aligned}$$

□

We can now show that $[\![-]\!]_w$ takes an infinite path of the atomic calculus to an infinite path in the weakening calculus.

Theorem 4.2.3 ($[\![-]\!]_w$ preserves non-termination). Let $T \in \Lambda\mu S_a$. If $(\!|\, T \,|\!)$ has an infinite reduction path, then $[\![\, (\!|\, T \,|\!) \,]\!]_w = (\!|\, T \,|\!)_w \in \Lambda\mu S_w$ has an infinite reduction path.

$$\begin{array}{ccc} (\!|\, T \,|\!) \in \Lambda\mu S_a & & [\![\, (\!|\, T \,|\!) \,]\!]_w = (\!|\, T \,|\!)_w \in \Lambda\mu S_w \\ \Big\downarrow & \overset{4.2.3}{\Longrightarrow} & \Big\downarrow \\ \infty & & \infty \end{array}$$

Proof. Let $T \in \Lambda\mu S$. Suppose that $(\!|\, T \,|\!)$ has an infinite reduction path. Since sharing

reductions are strongly normalizing, there must be an infinite (β, μ) path from $(\!|T|\!)$. By Lemma 4.2.1, if $(\!|T|\!) \longrightarrow_{\beta,\mu} t'$, then $[\![(\!| T |\!)]\!]_w \longrightarrow^+_{\beta,\mu} [\![t']\!]_w$. From $[\![t']\!]_w$, by Lemma 4.2.2, we know that one (β, μ) step in the atomic calculus corresponds to at least one (β, μ) step in the weakening calculus. Therefore $[\![(\!| T |\!)]\!]_w$ has a (β, μ) infinite path. □

4.3 PSN for $\Lambda\mu S_w$: exhaustive strategy

In the atomic calculus, reductions can occur inside weakenings, and in particular infinite β or μ-reductions can occur. The construction of an infinite reduction path in the $\Lambda\mu S$-calculus from an atomic infinite reduction path thus becomes difficult. Reductions inside weakenings correspond to zero steps in the $\Lambda\mu S$-calculus, therefore an atomic term t that is not strongly normalizing does not imply that its interpretation $[\![t]\!]$ does not terminate. For instance, take $x[\leftarrow \Omega] \in \Lambda\mu S_a$, where Ω is the usual diverging term. This term has an infinite path, but its interpretation $[\![x[\leftarrow \Omega]]\!] = x \in \Lambda\mu S$ is in normal form. We then must show that this problem doesn't occur with atomic terms which are translations $(\!| T |\!)$ of $\Lambda\mu S$-terms T. In particular, we can show that if we end up with an infinite reduction inside a weakening in the atomic calculus, we could have kept this reduction outside the weakening as well, hence also in the $\Lambda\mu S$-calculus. In our example, $x[\leftarrow \Omega]$ comes from reducing $(\lambda y.x[\leftarrow y])\Omega$, in which the infinite reduction on Ω could have stayed outside. Proving this aspect in the atomic calculus would be quite intricate because of the many syntactic constructs, so we will use the $\Lambda\mu S$-calculus with only explicit weakenings to show this.

In this section, for the sake of simplicity, expressions of the weakening calculus are referred to as *weakening terms* $T \in \Lambda\mu S_w$.

Recall that PSN is split into two parts. We have shown in Theorem 4.2.3 that $[\![-]\!]_w$ preserves non-termination. The other part is shown here, and requires a different proof. Unfortunately, we cannot prove that one reduction step in the weakening calculus $\Lambda\mu S_w$ corresponds to at least one step in the $\Lambda\mu S$-calculus. There are several obstacles coming from reductions inside weakenings. We thus work on weakening translations $(\!| T |\!)_w$ of $\Lambda\mu S$-terms. However, we still cannot directly construct a reduction in the $\Lambda\mu S$-calculus:

$$T \longrightarrow T_1 \longrightarrow T_2 \longrightarrow \dots$$

from one in the weakening calculus:

$$(\!| T |\!)_w \longrightarrow w_1 \longrightarrow w_2 \longrightarrow \dots$$

since again, reductions inside weakenings can happen for some w_n. To solve that problem, we show that if we have an infinite path:

$$(\!| T |\!)_w \longrightarrow w_1 \longrightarrow w_2 \longrightarrow \dots$$

we can choose an alternative path:

$$(\!| T |\!)_w \longrightarrow p_1 \longrightarrow p_2 \longrightarrow \dots$$

following a different strategy. We first use an important result (Corollary 4.3.10) in the weakening calculus, stating that a weakening term is weakly β/μ-normalizing if and only if it is β/μ-strongly normalizing. That is, without garbage disposal (i.e. weakening removals), we cannot avoid infinite reductions. From there, we design an *exhaustive* strategy that necessarily gives an infinite path if it exists, and keeps infinite reductions outside of weakenings. Then interpreting back in the $\Lambda\mu S$-calculus, we get:

$$T = \lfloor (\!| T |\!)_w \rfloor \longrightarrow \lfloor p_1 \rfloor \longrightarrow \lfloor p_2 \rfloor \longrightarrow \dots$$

The strategy we use is close to call-by-value, evaluating the argument first. Now if a $\Lambda\mu S$-term T is strongly normalizing, we can show that the exhaustive strategy of $(\!| T |\!)_w$ terminates.

$$\begin{array}{ccc}
(\!| T |\!)_w \in \Lambda\mu S_w & \xleftarrow[(\!|-|\!)_w]{} & T \in \Lambda\mu S \\
\Big\downarrow{\scriptstyle \beta,\mu}+ & & \Big\downarrow{\scriptstyle \beta,\mu} \\
(\!| T_1 |\!)_w \in \Lambda\mu S_w & & T_1 \\
\Big\downarrow & & \Big\downarrow \\
\dots & & \dots
\end{array}$$

We show in Section 5.2.1 that $\longrightarrow_w$ is strongly normalizing. As a consequence, if a weakening term $T \in \Lambda\mu S_w$ has an infinite reduction path, the path looks like:

$$T \longrightarrow_{\mu,\beta} T_1' \longrightarrow_w^* T_1 \longrightarrow_{\mu,\beta} \dots$$

with a finite number of w-steps between an infinite number of β or μ-steps. Thus, if $T \in \Lambda\mu S_w$ has an infinite reduction path, it has an infinite β, μ reduction path.

Each redex duplication produces a finite number of copies. To keep track of the duplications from $T \in \Lambda\mu S_w$, we annotate T with *labelings*. These labelings will be used to

show that for a weakening term that has finite and infinite reductions, normalization can only happen from weakening reductions.

Definition 4.3.1. [Labeling] Let $T \in \Lambda\mu S_w$. A *labeling* of T puts a $^\bullet$ marker on an arbitrary number of β/μ-redexes in T, which are written $(\mathcal{A}x.M)^\bullet N$.

We then evaluate the labeled redexes and inductively obtain a *labeled reduct* of an expression $T \in \Lambda\mu S_w$:

Definition 4.3.2. [Labeled reduct] A *labeled reduct* $T{\downarrow}$ of $T \in \Lambda\mu S_w$ is obtained as follows:

$$\begin{aligned}
\chi{\downarrow} &= \chi \\
(\mathcal{A}x.M){\downarrow} &= \mathcal{A}x.M{\downarrow} \\
((\lambda x.M)^\bullet N){\downarrow} &= M{\downarrow}\{N{\downarrow}/x\} \\
((\lambda x.M)^\bullet (N \circ \mathcal{S})){\downarrow} &= (M{\downarrow}\{N{\downarrow}/x\})\mathcal{S}{\downarrow} \\
((\mu\beta.M)^\bullet N){\downarrow} &= \mu\beta.M{\downarrow}\{(N{\downarrow} \circ \beta)/\beta\} \\
((\mu\beta.M)^\bullet \mathcal{S}){\downarrow} &= M{\downarrow}\{\mathcal{S}{\downarrow}/\beta\} \\
(M[\leftarrow N]){\downarrow} &= M{\downarrow}[\leftarrow N{\downarrow}] \\
@(M,N){\downarrow} &= @(M{\downarrow}, N{\downarrow}) \text{ if } M \text{ is unlabeled}
\end{aligned}$$

Remark 4.3.3. Inductively, we have $(M\{N/\chi\}){\downarrow} = (M{\downarrow}\{N{\downarrow}/\chi\})$.

The translation from M to its labeled reduct $M{\downarrow}$ can be seen as a *parallel reduction* step, which consists of a finite number of β, μ reductions on the labeled redexes.

Definition 4.3.4. The *parallel reduction* step $M \longrightarrow^\bullet_{\beta,\mu} M{\downarrow}$ reduces $M \in \Lambda\mu S_w$ to its labeled reduct $M{\downarrow}$.

We now show that if $M \in \Lambda\mu S_w$ is labeled, and $M \longrightarrow_{\beta,\mu} N$, then its labeled reduct $M{\downarrow}$ eventually reduces to $N{\downarrow}$.

Lemma 4.3.5. Let $M \in \Lambda\mu S_w$ be labeled. If $M \longrightarrow_{\beta,\mu} N$ and the redex that has been reduced is labeled, then $M{\downarrow} = N{\downarrow}$. Otherwise $M{\downarrow} \longrightarrow^+_{\beta,\mu} N{\downarrow}$.

Proof. By induction on the reduction $M \longrightarrow_{\beta,\mu} N$.

- $M = (\lambda x.T)U \longrightarrow_{\beta,\mu} N = T\{U/x\}$:

- If $M = (\lambda x.T)^\bullet U \longrightarrow_{\beta,\mu} N = T\{U/x\}$, then:

$$M{\downarrow} = ((\lambda x.T)^\bullet U){\downarrow} = T{\downarrow}\{U{\downarrow}/x\}$$

  Then, $N{\downarrow} = (T\{U/x\}){\downarrow} = T{\downarrow}\{U{\downarrow}/x\} = M{\downarrow}$.
- Otherwise, $M{\downarrow} = (\lambda x.T{\downarrow})U{\downarrow} \longrightarrow_{\beta,\mu} T{\downarrow}\{U{\downarrow}/x\} = N{\downarrow}$.

- $M = (\lambda x.T)(U \circ \mathcal{S}) \longrightarrow_{\beta,\mu} N = (T\{U/x\})\mathcal{S}$:
  - If $M = (\lambda x.T)^\bullet(U \circ \mathcal{S}) \longrightarrow_{\beta,\mu} N = (T\{U/x\})\mathcal{S}$, then:

$$M{\downarrow} = ((\lambda x.T)^\bullet(U \circ \mathcal{S})){\downarrow} = (T{\downarrow}\{U{\downarrow}/x\})\mathcal{S}{\downarrow}$$

    Then $N{\downarrow} = ((T\{U/x\})\mathcal{S}){\downarrow} = (T{\downarrow}\{U{\downarrow}/x\})\mathcal{S}{\downarrow} = M{\downarrow}$.
  - Otherwise, $M{\downarrow} = (\lambda x.T{\downarrow})(U{\downarrow} \circ \mathcal{S}{\downarrow}) \longrightarrow_{\beta,\mu} (T{\downarrow}\{U{\downarrow}/x\})\mathcal{S}{\downarrow} = N{\downarrow}$.

- $M = (\mu\beta.T)U \longrightarrow_{\beta,\mu} N = \mu\beta.T\{(U \circ \beta)/\beta\}$
  - If $M = (\mu\beta.T)^\bullet U \longrightarrow_{\beta,\mu} N = \mu\beta.T\{(U \circ \beta)/\beta\}$, then:

$$M{\downarrow} = ((\mu\beta.T)^\bullet U){\downarrow} = \mu\beta.T{\downarrow}\{(U{\downarrow} \circ \beta)/\beta\}$$

    Then $N{\downarrow} = (\mu\beta.T\{(U \circ \beta)/\beta\}){\downarrow} = \mu\beta.T{\downarrow}\{(U{\downarrow} \circ \beta)/\beta\} = M{\downarrow}$.
  - Otherwise, $M{\downarrow} = (\mu\beta.T{\downarrow})U{\downarrow} \longrightarrow_{\beta,\mu} \mu\beta.T{\downarrow}\{(U{\downarrow} \circ \beta)/\beta\} = N{\downarrow}$.

- $M = (\mu\beta.T)\mathcal{S} \longrightarrow_{\beta,\mu} N = T\{\mathcal{S}/\beta\}$
  - If $M = (\mu\beta.T)^\bullet\mathcal{S} \longrightarrow_{\beta,\mu} N = T\{\mathcal{S}/\beta\}$:

$$M{\downarrow} = ((\mu\beta.T)^\bullet\mathcal{S}){\downarrow} = T{\downarrow}\{\mathcal{S}{\downarrow}/\beta\}$$

    Then $N{\downarrow} = (T\{\mathcal{S}/\beta\}){\downarrow} = T{\downarrow}\{\mathcal{S}{\downarrow}/\beta\}$.
  - Otherwise, $M{\downarrow} = (\mu\beta.T{\downarrow})\mathcal{S}{\downarrow} \longrightarrow_{\beta,\mu} T{\downarrow}\{\mathcal{S}{\downarrow}/\beta\} = N{\downarrow}$.

- If $M = \mathcal{A}x.T \longrightarrow_{\beta,\mu} N$, then $N = \mathcal{A}x.T'$ where $T \longrightarrow_{\beta,\mu} T'$. By induction hypothesis, either the reduced redex is labeled and $T{\downarrow} = T'{\downarrow}$ or $T{\downarrow} \longrightarrow^+_{\beta,\mu} T'{\downarrow}$. By definition, $M{\downarrow} = \mathcal{A}x.(T{\downarrow})$ and $N{\downarrow} = \mathcal{A}x.(T'{\downarrow})$, therefore we can conclude.

- If $M = @(T,U) \longrightarrow_{\beta,\mu} N = @(T',U')$ then either $M = @(T,U) \longrightarrow_{\beta,\mu} @(T',U)$ or $M = @(T,U) \longrightarrow_{\beta,\mu} @(T,U')$.
  - Suppose $M = @(T,U) \longrightarrow_{\beta,\mu} N = @(T',U)$. By induction hypothesis, either the reduced redex is labeled and $T{\downarrow} = T'{\downarrow}$ or $T{\downarrow} \longrightarrow^+_{\beta,\mu} T'{\downarrow}$.

* If $@(T,U)$ is unlabeled, then by definition $M = (@(T,U))\downarrow = @(T\downarrow, U\downarrow)$ and $N = (@(T',U))\downarrow = @(T'\downarrow, U\downarrow)$. By induction hypothesis, $T\downarrow \longrightarrow^{+}_{\beta,\mu} T'\downarrow$ and we can conclude.
* If $@(T,U)$ is labeled and $T = \lambda x.V$, then $T' = \lambda x.V'$ and $M\downarrow = V\downarrow\{U\downarrow/x\}$, also $N\downarrow = V'\downarrow\{U\downarrow/x\}$, and $M\downarrow \longrightarrow_{\beta,\mu} N\downarrow$.
* The other cases whenever M is an abstraction are similar.

– Suppose $M = @(T,U) \longrightarrow_{\beta,\mu} N = @(T,U')$.
* If $@(T,U)$ is unlabeled, then by definition $M = (@(T,U))\downarrow = @(T\downarrow, U\downarrow)$ and $N = (@(T,U'))\downarrow = @(T\downarrow, U'\downarrow)$. By induction hypothesis, $U\downarrow \longrightarrow^{+}_{\beta,\mu} U'\downarrow$ and we can conclude.
* If $@(T,U)$ is labeled and $T = \lambda x.V$, then $M\downarrow = V\downarrow\{U\downarrow/x\}$, also $N\downarrow = V\downarrow\{U'\downarrow/x\}$, and $M\downarrow \longrightarrow^{+}_{\beta,\mu} N$.
* The other cases whenever M is an abstraction are similar.

- Suppose $M = (T[\leftarrow U]) \longrightarrow_{\beta,\mu} N = (T'[\leftarrow U'])$.
– If $M = (T[\leftarrow U]) \longrightarrow_{\beta,\mu} N = (T'[\leftarrow U])$, we have $T \longrightarrow_{\beta,\mu} T'$. By induction hypothesis, either the reduced redex is labeled and $T\downarrow = T'\downarrow$ or $T\downarrow \longrightarrow^{+}_{\beta,\mu} T'\downarrow$. By definition, $M\downarrow = (T\downarrow[\leftarrow U\downarrow])$ and $N\downarrow = (T'\downarrow[\leftarrow U\downarrow])$, therefore we can conclude.
– $M = (T[\leftarrow U]) \longrightarrow_{\beta,\mu} N = (T[\leftarrow U'])$, we have $U \longrightarrow_{\beta,\mu} U'$. By induction hypothesis, either the reduced redex is labeled and $U\downarrow = U'\downarrow$ or $U\downarrow \longrightarrow^{+}_{\beta,\mu} U'\downarrow$. By definition, $M\downarrow = (T\downarrow[\leftarrow U\downarrow])$ and $N\downarrow = (T\downarrow[\leftarrow U'\downarrow])$, therefore we can conclude.

□

Let $M \in \Lambda\mu S_w$, such that M has an infinite reduction path. The previous lemma will be used to show that if M has a finite reduction path, it must come from weakening rules. It shows that using only β,μ-rules guarantees to find an infinite path if it exists, i.e. all β,μ reductions paths are infinite or *perpetual* in the sense of Barendregt [Bar84].

Lemma 4.3.6. Let $M \in \Lambda\mu S_w$. If M has an infinite β,μ reduction and $M \longrightarrow_{\beta,\mu} N$, then N has an infinite β,μ reduction.

Proof. Since $M \longrightarrow_{\beta,\mu} N$, we can label the redex taking M to N, i.e. $M\downarrow = N$. The infinite reduction path of M looks like:

$$M = M_0 \longrightarrow_{\beta,\mu} M_1 \longrightarrow_{\beta,\mu} M_2 \ldots$$

Using Lemma 4.3.5, we will show that the following path is infinite:

$$N = M_0\!\downdownarrows \longrightarrow^*_{\beta,\mu} M_1\!\downdownarrows \longrightarrow^*_{\beta,\mu} \dots$$

where $M_{i+1}\!\downdownarrows = M_i\!\downdownarrows$ if the reduced redex in $M_i \longrightarrow_{\beta,\mu} M_{i+1}$ is labeled, otherwise $M_i\!\downdownarrows \longrightarrow^+_{\beta,\mu} M_{i+1}\!\downdownarrows$. Now consider the parallel reduction step $M_i \longrightarrow^\bullet_{\beta,\mu} M_i\!\downdownarrows$. It has a finite number of steps, therefore there are a finite number of labeled reductions between two unlabeled reductions. We have the following path for M:

$$M = M_0 \longrightarrow_{\beta,\mu} M_1 \longrightarrow_{\beta,\mu} M_2 \dots$$

where $M_i\!\downdownarrows = M_{i+1}$ if the reduction $M_i \longrightarrow_{\beta,\mu} M_{i+1}$ is labeled, otherwise $M_i \longrightarrow^+_{\beta,\mu} M_{i+1}$. Since any sequence of labeled steps

$$M_i \longrightarrow_{\beta,\mu} M_{i+1} \longrightarrow_{\beta,\mu} \dots \longrightarrow_{\beta,\mu} M_{i+k}$$

must be finite, in the infinite path of M

$$M = M_0 \longrightarrow_{\beta,\mu} M_1 \longrightarrow_{\beta,\mu} M_2 \longrightarrow_{\beta,\mu} \dots$$

there must be an infinite number of unlabeled reductions. Therefore, the sequence

$$M\!\downdownarrows = M_0\!\downdownarrows \longrightarrow^*_{\beta,\mu} M_1\!\downdownarrows \longrightarrow^*_{\beta,\mu} \dots$$

has infinitely many sequences $M_i\!\downdownarrows \longrightarrow^+_{\beta,\mu} M_{i+1}\!\downdownarrows$, so it must be infinite. □

Example 4.3.7. Take $\Omega = (\lambda x.(x)x)\lambda x.(x)x \in \Lambda\mu S_w$, which reduces to itself:

$$\Omega \longrightarrow_\beta W_1 \longrightarrow_\beta W_2 \longrightarrow_\beta \cdots \longrightarrow_\beta W_k = \Omega \dots$$

Consider $M = (\lambda x.(\lambda y.y)^\bullet x)\Omega$, which has the reduction sequence:

$$\begin{aligned} M &\longrightarrow_\beta M_1 = (\lambda x.(\lambda y.y)^\bullet x)W_1 \\ &\longrightarrow_\beta M_2 = (\lambda y.y)^\bullet W_1 \\ &\longrightarrow_\beta M_3 = (\lambda y.y)^\bullet W_2 \\ &\longrightarrow_\beta M_4 = W_2 \\ &\longrightarrow_\beta M_5 = W_3 \dots \end{aligned}$$

Then $M \longrightarrow_\beta N = (\lambda x.x)\Omega = M\Downarrow$, and:

$$\begin{aligned} N &\longrightarrow_\beta M_1\Downarrow = (\lambda x.x)W_1 \\ &\longrightarrow_\beta M_2\Downarrow = W_1 \\ &\longrightarrow_\beta M_3\Downarrow = W_2 \\ &= \quad M_4\Downarrow = W_2 \\ &\longrightarrow_\beta M_5\Downarrow = W_3 \end{aligned}$$

Example 4.3.8. Take Ω and its reduction sequence from the previous example. Consider $M = (\lambda x.(x)x)((\lambda y.y)^\bullet\Omega)$, which has the reduction sequence:

$$\begin{aligned} M &\longrightarrow_\beta M_1 = (\lambda x.(x)x)((\lambda y.y)^\bullet W_1) \\ &\longrightarrow_\beta M_2 = ((\lambda y.y)^\bullet W_1)((\lambda y.y)^\bullet W_1) \\ &\longrightarrow_\beta M_3 = (W_1)((\lambda y.y)^\bullet W_1) \\ &\longrightarrow_\beta M_4 = (W_1)W_1 \\ &\longrightarrow_\beta M_5 = (W_2)W_1 \ldots \end{aligned}$$

Then $M \longrightarrow_\beta N = (\lambda x.x)\Omega = M\Downarrow$, and:

$$\begin{aligned} N &\longrightarrow_\beta M_1\Downarrow = (\lambda x.(x)x)W_1 \\ &\longrightarrow_\beta M_2\Downarrow = (W_1)W_1 \\ &= \quad M_3\Downarrow = (W_1)W_1 \\ &= \quad M_4\Downarrow = (W_1)W_1 \\ &\longrightarrow_\beta M_5\Downarrow = (W_2)W_1 \end{aligned}$$

Example 4.3.9. Let $T = (\lambda x.y[\leftarrow x])(\delta)\delta \in \Lambda\mu S_w$ where $\delta = \lambda x.(x)x$. Then T has an infinite reduction on $(\delta)\delta$, and reduces to T or $U = y[\leftarrow (\delta)\delta])$, which has an infinite reduction on $(\delta)\delta$.

The following corollary shows that a weakening term with an infinite reduction that is weakly normalizing can only normalize with weakening reductions.

Corollary 4.3.10. If $M \in \Lambda\mu S_w$ has an infinite β, μ reduction path, then no β, μ reduction path is finite.

Proof. Suppose by contradiction that $M \in \Lambda\mu S_w$ has an infinite β, μ reduction path, and that there is a finite β, μ reduction path from M:

$$M = M_0 \longrightarrow_{\beta,\mu} M_1 \ldots \longrightarrow_{\beta,\mu} M_n$$

Using Lemma 4.3.6, if M reduces to M_1, then M_1 has an infinite β, μ reduction. Inductively, M_{n-1} reduces to M_n which has an infinite reduction path, thus cannot be in normal form, which is a contradiction. □

Example 4.3.11. Considering $T = (\lambda x.y[\leftarrow x])(\delta)\delta \in \Lambda\mu S_w$, any β, μ reduction path will infinitely reduce $(\delta)\delta$.

We showed that if a weakening term has an infinite reduction, then any path only involving β, μ-reductions is infinite. We now want to define a particular exhaustive strategy in $\Lambda\mu S_w$ that never involves reductions inside weakenings. That way, translating back to the $\Lambda\mu S$-calculus, we get an infinite reduction path. Let $M \in \Lambda\mu S_w$ be a weakening term.

Definition 4.3.12. The *call-by-value-like exhaustive strategy* on $M \in \Lambda\mu S_w$ is the reduction sequence

$$M = M_1 \longrightarrow_{\beta,\mu} M_2 \longrightarrow_{\beta,\mu} M_3 \longrightarrow_{\beta,\mu} \ldots$$

where $M_{i+1} = \rho(M_i)$, such that

$$\rho(\bullet) = \bullet$$
$$\rho(\chi) = \chi$$
$$\rho(\mathcal{A}x.N) = \mathcal{A}x.\rho(N)$$
$$\rho(@(N,P)) = \begin{cases} @(N,\rho(P)) & \text{if } P \text{ not in NF} \\ T\{P/x\}[\Phi] & \text{if } N = (\lambda x.T)[\Phi] \\ & \text{and } P \text{ in NF} \\ ((T\{Q/x\})[\Phi])\mathcal{S} & \text{if } N = (\lambda x.T)[\Phi], P = Q \circ \mathcal{S}, \\ & \text{and } P \text{ in NF} \\ \mu\beta.T\{(P \circ \beta)/\alpha\}[\Phi] & \text{if } N = (\mu\alpha.T)[\Phi], \\ & \text{and } P \text{ in NF} \\ T\{\mathcal{S}/\alpha\}[\Phi] & \text{if } N = (\mu\alpha.T)[\Phi], P = \mathcal{S}, \\ & \text{and } P \text{ in NF} \\ @(\rho(N),P) & \text{if } N \neq (\mathcal{A}x.T)[\Phi] \text{ and } P \text{ in NF} \end{cases}$$
$$\rho(N[\leftarrow P]) = \begin{cases} \rho(N)[\leftarrow P] & \text{if } N \text{ not NF} \\ N[\leftarrow \rho(P)] & \text{otherwise} \end{cases}$$

such that M is in (ρ-)normal form (NF) if $\rho(M) = M$.

If we start from the translation of a lambda-mu term, with this exhaustive strategy, we never reduce inside weakenings. The typical situation we want to avoid is $(\lambda x.M[\leftarrow (x)\delta])\delta \longrightarrow_\beta M[\leftarrow \Omega]$. In the $\Lambda\mu S$-calculus $\lfloor M[\leftarrow \Omega]\rfloor = \lfloor M\rfloor$, so we would have an infinite path in the weakening calculus but not in the $\Lambda\mu S$-calculus. Since we start from translations of $\Lambda\mu S$-expressions, this situation (having weakenings other than $[\leftarrow \chi]$) cannot happen.

We show that using exhaustive reduction, the only kind of weakenings we obtain are either $[\leftarrow \chi]$ or $[\leftarrow T]$ where T cannot be reduced or help form a redex. In the latter case T is called *frozen*. We first define frozen abstraction constructors inside an expression M, which are constructors that remain untouched after all possible reductions.

Definition 4.3.13. An abstraction constructor $\mathcal{A}x$ in $M \in \Lambda\mu S_w$ is *frozen* if:

- $M = \mathcal{A}x.N$
- $M = \mathcal{A}y.N$ where $y \neq x$ and $\mathcal{A}x$ is frozen in N
- $M = @(N,P)$ and M is not a redex, N is normal, $\mathcal{A}x$ is frozen in N or P

- $M = N[\leftarrow P]$ and $\mathcal{A}x$ is frozen in N or P

Example 4.3.14. Consider λy in $M = ((\lambda z.\lambda k.(k)z)x)\lambda y.y$. The term M is not a redex, but reduces to $(\lambda k.(k)x)\lambda y.y$, then λy participates in a redex, therefore is not frozen.

Definition 4.3.15. A variable χ in M is *frozen* if $\mathcal{A}\chi$ is a frozen abstraction of M. A subexpression $N \neq \chi$ of M is *frozen* if its variables are either free or frozen, and if it is in normal form.

Example 4.3.16. Let $\delta = \lambda y.(y)y$. Consider $N = (\lambda x.M[\leftarrow (x)\delta])$. λx is frozen in N, therefore x is frozen in N. Also, λy is frozen in δ, so y is frozen in N. Therefore $(x)\delta$ is frozen in N.

If an abstraction $\mathcal{A}x$ is frozen in M, its bound variable x remains untouched and never participates in a redex. Therefore, if an expression has no redexes and only free or frozen variables, it can never become or participate in a redex either. We now show that with exhaustive reductions, only frozen expressions or variables can appear inside weakenings.

Definition 4.3.17. Let $M \in \Lambda\mu S_w$. We say that M is *cool* if for any weakening $[\leftarrow F]$ in M, either $F = \chi$ or $F \neq \chi$ is frozen.

As stated before, with the translation of a $\Lambda\mu S$-expression, only weakenings of the form $[\leftarrow \chi]$ appear, therefore for any $T \in \Lambda\mu S$, its translation $(\!|\, T \,|\!)_w$ is cool. We now show that from $(\!|\, T \,|\!)_w$, applying exhaustive reduction preserves coolness.

Lemma 4.3.18. Let $M \in \Lambda\mu S_w$. If M is cool, then $\rho(M)$ is cool.

Proof. Let $M \in \mathbb{T}, \tau \in \mathbb{T} \cup \mathbb{S}$.

1. $\bullet = \rho(\bullet), \chi = \rho(\chi)$ are cool and in normal form.
2. Let $\mathcal{A}x.M$ be cool. Then $\rho(\mathcal{A}x.M) = \mathcal{A}x.\rho(M)$. By definition M is cool, and by induction hypothesis $\rho(M)$ is cool, thus $\rho(\mathcal{A}x.M)$ is cool.
3. Let $M = @(N, \tau)$ be cool. By definition, N and τ are cool.
   - If $N = (\lambda x.T)[\Phi]$, and τ is in normal form, then $\rho(M) = T\{\tau/x\}[\Phi]$.
     (a) Since M is cool, and the abstraction λx is not frozen, for every weakening $[\leftarrow \tau']$ in M we have either $x = \tau'$ or $x \notin FV(\tau')$. We now show that τ is frozen.

- If τ is a variable, it is frozen.
- If there are only free variables in τ, then τ is frozen.
- If χ' is a bound variable of τ, it is bound to an abstraction $\mathcal{A}\chi'$, which is frozen.

Therefore, τ is frozen. Therefore, any weakening in $\rho(M)$ is either a weakening in M or $[\leftarrow \tau]$ which is frozen, then $\rho(M)$ is cool.

(b) Otherwise, there are no new weakenings formed, so $\rho(M)$ stays cool.

- The reasoning is similar for other redexes.
- In the remaining cases for applications, we have $\rho(@(N,\tau)) = @(\rho(N),\tau)$ and $\rho(@(N,\tau)) = @(N,\rho(\tau))$, then we can conclude by induction hypothesis.

4. Let $M = N[\leftarrow \tau]$ be cool. By definition, N and τ are cool, and τ is frozen.
   - $\rho(M) = \rho(N)[\leftarrow \tau]$ if N is not in normal form. By induction, $\rho(N)$ is cool, therefore $\rho(M)$ is cool.
   - $\rho(M) = N[\leftarrow \rho(\tau)]$ otherwise. By induction $\rho(\tau)$ is cool. Also, τ is frozen i.e. each variable is either free or bound to a frozen abstraction, and that is preserved after applying $\rho(\tau)$. Note that since τ is frozen (M being cool), it is in normal form and $\rho(\tau) = \tau$.

□

The following lemma, along with Lemma 4.1.5, shows that the correspondence between β,μ-steps in $\Lambda\mu S_a, \Lambda\mu S_w, \Lambda\mu S$ works with the interpretation functions $[\![-]\!], [\![-]\!]_w, \lfloor - \rfloor$, but not with the translations $(\!|-|\!), (\!|-|\!)_w$.

Lemma 4.3.19. For $t \in \Lambda\mu S_a$, $\lfloor [\![t]\!]_w \rfloor = [\![t]\!]$.

$$\begin{array}{ccc} t \in \Lambda\mu S_a & \xrightarrow{[\![-]\!]_w} & [\![t]\!]_w \in \Lambda\mu S_w \\ \Big\downarrow{\scriptstyle [\![-]\!]} & & \Big\downarrow{\scriptstyle \lfloor - \rfloor} \\ [\![t]\!] \in \Lambda\mu S & = & \lfloor [\![t]\!]_w \rfloor \in \Lambda\mu S \end{array}$$

Proof. Let χ be a variable, τ be a term or a stream. By induction on t.

- If $t = \chi$, then:

$$\lfloor [\![\chi]\!]_w \rfloor = \chi = [\![\chi]\!]$$

- If $t = \mathcal{A}x.u$, then:

$$\begin{aligned}\lfloor [\![\mathcal{A}x.u]\!]_w \rfloor &= \mathcal{A}x.\lfloor [\![u]\!]_w \rfloor \\ &= \mathcal{A}x.[\![u]\!] \\ &= [\![\mathcal{A}x.u]\!]\end{aligned}$$

- If $t = @(u, \tau)$, then:

$$\begin{aligned}\lfloor [\![@(u,\tau)]\!]_w \rfloor &= \lfloor @([\![t]\!]_w, [\![\tau]\!]_w) \rfloor \\ &= @(\lfloor [\![t]\!]_w \rfloor, \lfloor [\![\tau]\!]_w \rfloor) \\ &= @([\![t]\!], [\![\tau]\!]) \\ &= [\![@(t,\tau)]\!]\end{aligned}$$

- If $t = u^*[\phi]$, then:

$$\begin{aligned}\lfloor [\![u^*[\phi]]\!]_w \rfloor &= \lfloor [\![u^*]\!]_w \{\!\!\{ \phi \}\!\!\}_w \rfloor \\ &= \lfloor [\![u^*]\!]_w \rfloor \lfloor \{\!\!\{ \phi \}\!\!\}_w \rfloor \\ &= [\![u^*]\!] \lfloor \{\!\!\{ \phi \}\!\!\}_w \rfloor \\ &= [\![u^*]\!] \{\!\!\{ \phi \}\!\!\} \\ &= [\![u^*[\phi]]\!]\end{aligned}$$

- For sharings:

$$\lfloor \{\!\!\{ \vec{\chi_p} \leftarrow \tau \}\!\!\}_w \rfloor = \begin{cases} \lfloor [\leftarrow [\![\tau]\!]_w] \rfloor = \emptyset = \{\!\!\{ \leftarrow \tau \}\!\!\} & \text{if } p = 0 \\ \\ \lfloor \{ [\![\tau]\!]_w / \chi_i \}_{1 \le i \le p} \rfloor = \{ \lfloor [\![\tau]\!]_w \rfloor / \chi_i \}_{1 \le i \le p} & \\ = \{ [\![\tau]\!] / \chi_i \}_{1 \le i \le p} = \{\!\!\{ \vec{\chi_p} \leftarrow \tau \}\!\!\} & \text{if } p \ge 1 \end{cases}$$

•

$$\lfloor \{\!|\, x_p^{\vec{}} \twoheadleftarrow \mathcal{A}y.\langle\, t_p^{\vec{}} \,\rangle[\Psi] \,|\!\}_w \rfloor = \begin{cases} \lfloor \{\!|\, \Psi \,|\!\}_w \{\bullet/y\} \rfloor = \lfloor \{\!|\, \Psi \,|\!\}_w \rfloor \{\bullet/y\} \\ = \{\!|\, \Psi \,|\!\}\{\bullet/y\} & \text{if } p = 0 \\ \\ \lfloor \{\mathcal{A}y.[\![\, t_i[\Psi] \,]\!]_w/x_i\}_{1\leq i\leq p} \rfloor \\ = \{\mathcal{A}y.\lfloor [\![\, t_i[\Psi] \,]\!]_w \rfloor/x_i\}_{1\leq i\leq p} \\ = \{\mathcal{A}y.[\![\, t_i[\Psi] \,]\!]/x_i\}_{1\leq i\leq p} \\ = \{[\![\, \mathcal{A}y.t_i[\Psi] \,]\!]/x_i\}_{1\leq i\leq p} & \text{if } p \geq 1 \end{cases}$$

□

We can now conclude with PSN for the weakening calculus:

Theorem 4.3.20 (PSN for the weakening calculus). Let $T \in \Lambda\mu S$. If $(\!|\, T \,|\!)_w$ has an infinite reduction path, then T has an infinite reduction path.

$$\begin{array}{ccc} (\!|\, T \,|\!)_w \in \Lambda\mu S_w & & T \in \Lambda\mu S \\ \Big\downarrow & \overset{4.3.20}{\Longrightarrow} & \Big\downarrow \\ \infty & & \infty \end{array}$$

Proof. We work on the contrapositive. We suppose that for $T \in \Lambda\mu S$, its translation $(\!|\, T \,|\!)_w \in \Lambda\mu S_w$ has an infinite reduction. Since weakening reductions are strongly normalizing (Theorem 5.2.8), we can skip them, i.e. $(\!|T|\!)_w$ has an infinite β, μ reduction path. Then starting from $(\!|T|\!)_w$, we always get an infinite reduction with β, μ reductions, by Corollary 4.3.10.

Going back to $\Lambda\mu S$, we always have:

$$\begin{array}{ccc} M \in \Lambda\mu S_w & \longrightarrow & \lfloor M \rfloor \in \Lambda\mu S \\ \Big\downarrow{\scriptstyle \beta/\mu} & & \Big\downarrow{\scriptstyle (\beta/\mu)^*} \\ N \in \Lambda\mu S_w & \longrightarrow & \lfloor N \rfloor \in \Lambda\mu S \end{array}$$

where $\lfloor M \rfloor = \lfloor N \rfloor$ if $M \longrightarrow_{\beta,\mu} N$ inside a weakening, and $\lfloor M \rfloor \longrightarrow_{\beta,\mu} \lfloor N \rfloor$ otherwise. Using the exhaustive strategy, we get a particular infinite path, which consists of only cool terms. Therefore, we get a path where reductions never happen inside weakenings.

In the weakening calculus, for a step $M \longrightarrow_{\beta,\mu} N$ that is not inside a weakening:

$$\begin{array}{ccc} M \in \Lambda\mu S_w & \longrightarrow & \lfloor M \rfloor \in \Lambda\mu S \\ \Big\downarrow{\scriptstyle \beta/\mu} & & \Big\downarrow{\scriptstyle \beta/\mu} \\ N \in \Lambda\mu S_w & \longrightarrow & \lfloor N \rfloor \in \Lambda\mu S \end{array}$$

Going back to $\Lambda\mu S$, using $\lfloor (\!| T |\!)_w \rfloor = T$ by Lemma 4.3.19, we get an infinite path from T. □

Chapter 5

Strong normalization of sharing reductions

Strong normalization of sharing reductions is the other core lemma to prove preservation of strong normalization, stating that solely β, μ reductions are responsible for divergence. To show that $\longrightarrow_s$ is strongly normalizing, we construct a measure that strictly decreases after each reduction. Several parameters can decrease during the reduction:

- A reduction concerns a weakening reduction, which is strongly normalizing since terms become strictly smaller.
- The *weight* of subexpressions of a term, which gathers multisets representing the number of copies of subexpressions.
- The number of closures.
- The *depths* of subexpressions of a term, which gathers multisets representing the paths to the closures inside the term.

5.1 Preliminaries on multisets

Definition 5.1.1. [Multiset] A *multiset* m on a set A is a function

$$\begin{aligned} m : A &\to \mathbb{N} \\ a &\mapsto m(a) \end{aligned}$$

Remark 5.1.2.

1. The multiset m on A can be seen as a set where elements of A may appear several times. For each element $a \in A$, $m(a)$ denotes the number of occurrences (or *multiplicity*) of a.
2. As with any function, it is also possible to define m with its graph, a set $\{a^{m(a)} \mid a \in A\}$, where we omit a^0.

Example 5.1.3. Looking at the integer factorization of 120, we have $120 = 2*2*2*3*5$, which can be represented (following the second notation) by the multiset $\{2^3, 3^1, 5^1\} = \{2^3, 3, 5\}$.

Remark 5.1.4. The *empty multiset* is written $\emptyset$.

Definition 5.1.5. The *sum* (or *union*) $f + g$ of two functions f and g with the same domain A is the function

$$\begin{aligned} f + g : A &\to \mathbb{N} \\ a &\mapsto f(a) + g(a) \end{aligned}$$

equivalently $f + g = \{a^{f(a)+g(a)} \mid a \in A\}$.

Remark 5.1.6. This definition extends to multisets $m, n : A \to \mathbb{N}$.

Example 5.1.7. The union $\{2^3, 3, 5\} + \{3^3\}$ gives $\{2^3, 3^4, 5\}$.

Definition 5.1.8. The *inclusion* relation between two multisets m, n on $A \to \mathbb{N}$ is defined by:

$$m \subseteq n \text{ if for any } x \in A,\ m(x) \leq n(x)$$

Definition 5.1.9. If A can be equipped with a partial (strict) order $<_A$, an order (as in [JL82]) can be defined for multisets over A: $m \prec n$ if:

1. $m \neq n$
2. $\forall x \in A.\, m(x) > n(x) \implies \exists z \in A.\, z >_A x \wedge m(z) < n(z)$.

Definition 5.1.10. A *finite* multiset m is a multiset where there is a finite number of elements $a \in A$ such that $m(a) \neq 0$. We denote by $\mathcal{M}_f(A)$ the collection of finite multisets over A.

For a multiset $f : A \to \mathbb{N}$, and $x_1, \ldots, x_p \in A$, we denote by $f \backslash x_1, \ldots, x_p$ the multiset f with all copies of each x_i removed:

$$(f \backslash x_1, \ldots, x_p)(y) = \begin{cases} 0 & \text{if } y = x_i \text{ for some } 1 \leq i \leq p \\ f(y) & \text{otherwise} \end{cases}$$

5.2 Weakening reduction as a measure

The following theorem shows that a sharing reduction corresponds to zero or more steps in the weakening calculus. We will show the strong normalization of weakening reductions $\longrightarrow_w$, along with this theorem, to construct a strictly decreasing measure for the strong normalization of sharing reductions, the first parameter of the measure being a reduction occurring in the weakening calculus.

Theorem 5.2.1 ($[\![-]\!]_w$ commutes with $\longrightarrow_s$). Let $s^*, t^* \in \Lambda\mu S_a$ such that $s^* \longrightarrow_s t^*$. Then $[\![s^*]\!]_w \longrightarrow_w^* [\![t^*]\!]_w$.

Remark 5.2.2. Here, substitution and parallel substitutions are the same since all occurrences of variables are linear. Let $\tau \in \mathbb{T} \cup \mathbb{S}$ be a term or a stream, let $\chi_i \in \mathbb{T} \cup \mathbb{S}$ be variables.

Whenever a reduction rule in the atomic calculus does not involve a weakening, the translation in the weakening calculus corresponds to that in the $\Lambda\mu S$-calculus, thus a s-reduction corresponds to 0 weakening reductions.

Proof. We show the correspondence for each rule.

Compounding rules

$$\begin{aligned} [\leftarrow \chi_i][\chi_1, \ldots, \chi_q \leftarrow \tau] &\longrightarrow_s [\chi_1, \ldots, \chi_{i-1}, \chi_{i+1}, \ldots, \chi_q \leftarrow \tau] \\ \{\!| [\leftarrow \chi_i][\chi_1, \ldots, \chi_q \leftarrow \tau] |\!\}_w &= [\leftarrow \chi_i]\{[\![\tau]\!]_w / \vec{\chi_q}\} \\ &= [\leftarrow [\![\tau]\!]_w]\{[\![\tau]\!]_w / \chi_1, \ldots, \chi_{i-1}, \chi_{i+1}, \ldots, \chi_q\} \\ &\longrightarrow_w \{\!| \chi_1, \ldots, \chi_{i-1}, \chi_{i+1}, \ldots, \chi_q \leftarrow \tau |\!\}_w \end{aligned}$$

since $[\![\tau]\!]_w$ becomes a subexpression.

Lifting rules

1. Lifting inside abstractions (similar for applications)

(a)

$$\begin{aligned}
\mathcal{A}x.(u[\leftarrow \tau]) &\longrightarrow_s (\mathcal{A}x.u)[\leftarrow \tau] \\
[\![\, \mathcal{A}x.(u[\leftarrow \tau]) \,]\!]_w &= \mathcal{A}x.([\![\, u \,]\!]_w[\leftarrow [\![\, \tau \,]\!]_w]) \\
&\longrightarrow_w (\mathcal{A}x.[\![\, u \,]\!]_w)[\leftarrow \tau] \\
&= [\![\, (\mathcal{A}x.u)[\leftarrow \tau] \,]\!]_w
\end{aligned}$$

(b)

$$\begin{aligned}
\mathcal{A}x.(u[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]) &\longrightarrow_s (\mathcal{A}x.u)[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]] \\
[\![\, \mathcal{A}x.(u[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]) \,]\!]_w &= \mathcal{A}x.([\![\, u \,]\!]_w\{\!|\ \Psi\ |\!\}_w\{\bullet/y\}) \\
&= [\![\, \mathcal{A}x.u \,]\!]_w\{\!|\ \Psi\ |\!\}_w\{\bullet/y\} \\
&= [\![\, (\mathcal{A}x.u)[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]] \,]\!]_w
\end{aligned}$$

2. Lifting sharings inside sharings

(a)

$$\begin{aligned}
[\leftarrow \tau[\leftarrow \tau']] &\longrightarrow_s [\leftarrow \tau][\leftarrow \tau'] \\
[\leftarrow [\![\, \tau \,]\!]_w[\leftarrow [\![\, \tau' \,]\!]_w]] &\longrightarrow_w [\leftarrow [\![\, \tau \,]\!]_w][\leftarrow [\![\, \tau' \,]\!]_w]
\end{aligned}$$

(b) Consider the reduction

$$t^*[\vec{\chi_p} \leftarrow s^*[\leftarrow u^*]] \longrightarrow_s t^*[\vec{\chi_p} \leftarrow s^*][\leftarrow u^*]$$

Let

$$T^* = [\![\, t^* \,]\!]_w, S^* = [\![\, s^* \,]\!]_w, U^* = [\![\, u^* \,]\!]_w$$

Then we have

$$\{\!|\ [\vec{\chi_p} \leftarrow s^*[\leftarrow u^*]]\ |\!\}_w = \{S^*[\leftarrow U^*]/\chi_i\}_{i \leq p}$$

where $[\leftarrow U^*]$ appears p times inside T^*, whereas

$$\{\!|\ [\vec{\chi_p} \leftarrow s^*][\leftarrow u^*]\ |\!\}_w = \{S^*/\chi_i\}_{i \leq p}[\leftarrow U^*]$$

where $[\leftarrow U^*]$ appears once at top-level. Since t^* (resp. T^*) does not bind variables in u^* (resp. U^*), lifting rules can be applied to $[\leftarrow u^*]$ (resp.

$[\leftarrow U^*]$), and these sharings can be lifted to the top-level. In the weakening calculus, we have $T^*\{S^*[\leftarrow U^*]/\chi_i\}_{i\leq p} \longrightarrow_w^* T^*\{S^*/\chi_i\}\underbrace{[\leftarrow U^*]\ldots[\leftarrow U^*]}_{n}$. Since U^* appears as a subterm, we can remove all weakenings $[\leftarrow U^*]$ except the first one. It is possible to have 0 weakening steps, for instance consider $\chi[\chi \leftarrow s^*[\leftarrow u^*]]$.

$$\begin{aligned}
[\vec{\chi_p} \leftarrow \tau[\leftarrow \tau']] &\longrightarrow_s [\vec{\chi_p} \leftarrow \tau][\leftarrow \tau'] \\
\{\!| \vec{\chi_p} \leftarrow \tau[\leftarrow \tau'] |\!\}_w &= \{[\![\tau[\leftarrow \tau']]\!]_w/\vec{\chi_p}\} \\
&\longrightarrow_w^* \{[\![\tau]\!]_w/\vec{\chi_p}\}[\leftarrow [\![\tau']\!]_w] \\
&= \{\!| [\vec{\chi_p} \leftarrow \tau][\leftarrow \tau'] |\!\}_w
\end{aligned}$$

since τ' becomes a subterm.

3. Lifting distributors inside sharings

   (a)

$$\begin{aligned}
[\leftarrow \tau[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]] &\longrightarrow_s [\leftarrow \tau][\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]] \\
\{\!| [\leftarrow \tau[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]] |\!\}_w &= [\leftarrow [\![\tau]\!]_w \{\!| \Psi |\!\}_w\{\bullet/y\}] \\
&= [\leftarrow [\![\tau]\!]_w]\{\!| \Psi |\!\}_w\{\bullet/y\} \\
&= \{\!| [\leftarrow [\![\tau]\!]_w][\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]] |\!\}_w
\end{aligned}$$

   (b) In this case, we get p copies of $\{\!|\Psi|\!\}_w$ that we can annotate $\{\!|\Psi_1|\!\}_w, \ldots, \{\!|\Psi_p|\!\}_w$ on the LHS, and one copy on the RHS. Substitutions can occur inside weakenings, so we consider each $\{\!| \Psi_i |\!\}_w$, from p to 1. We proceed as follows:

      i. if $\{\!| \Psi_i |\!\}_w$ is a substitution, then LHS and RHS are equal
      ii. if $\{\!| \Psi_i |\!\}_w$ is a weakening $[\leftarrow U^*]$, we need to lift all instances of $[\leftarrow U^*]$ to the top-level, then delete any duplicates.

   Whenever we have a lifting from non-weakening closures, we apply a similar reasoning.

$$\begin{aligned}
[\vec{\chi_p} \leftarrow \tau[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]] &\longrightarrow_s [\vec{\chi_p} \leftarrow \tau][\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]] \\
\{([\![\tau]\!]_w\{\!| \Psi |\!\}_w\{\bullet/y\})/\vec{\chi_p}\} &\longrightarrow_w^* \{[\![\tau]\!]_w/\vec{\chi_p}\}\{\!| \Psi |\!\}_w\{\bullet/y\}
\end{aligned}$$

4. Lifting sharings inside distributors

(a)

$$[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\leftarrow \tau]] \longrightarrow_s [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\leftarrow \tau]$$
$$\begin{aligned}\{\!|\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\leftarrow \tau]|\!\}_w &= \{\!|\ \Psi\ |\!\}_w\{\bullet/y\}[\leftarrow [\![\ \tau\ ]\!]_w] \\ &= \{\!|\ [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\leftarrow \tau]\ |\!\}_w\end{aligned}$$

(b) Similar reasoning as other lifting cases from non-weakening closures.

$$[\vec{\chi_p} \twoheadleftarrow \mathcal{A}y.\langle\ \vec{\tau_p}\ \rangle[\Psi][\leftarrow \tau']] \longrightarrow_s [\vec{\chi_p} \twoheadleftarrow \mathcal{A}y.\langle\ \vec{\tau_p}\ \rangle[\Psi]][\leftarrow \tau']$$
$$\begin{aligned}&\{\!|\ \vec{\chi_p} \twoheadleftarrow \mathcal{A}y.\langle\ \vec{\tau_p}\ \rangle[\Psi][\leftarrow \tau']\ |\!\}_w = \\ &\qquad \{(\mathcal{A}y.[\![\ \tau_j\ ]\!]_w\{\!|\ \Psi\ |\!\}_w[\leftarrow [\![\ \tau'\ ]\!]_w])/\chi_j\}_{j\leq p} \\ &\qquad \longrightarrow^*_w \{(\mathcal{A}y.[\![\ \tau_j\ ]\!]_w\{\!|\ \Psi\ |\!\}_w)/\chi_j\}_{j\leq p}[\leftarrow [\![\ \tau'\ ]\!]_w]\end{aligned}$$

since τ' becomes a subexpression.

5. Lifting distributors inside distributors

(a)

$$[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]] \longrightarrow_s [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]$$
$$\begin{aligned}\{\!|\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]|\!\}_w &= \{\!|\ \Psi\ |\!\}_w\{\bullet/y\}\{\!|\ \Theta\ |\!\}_w\{\bullet/z\} \\ &= \{\!|\ [\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]\ |\!\}_w\end{aligned}$$

(b) Similar reasoning as other lifting cases from non-weakening closures.

$$\begin{aligned}&[\vec{\chi_p} \twoheadleftarrow \mathcal{A}y.\langle\ \vec{\tau_p}\ \rangle[\Psi][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]] \longrightarrow_s \\ &\qquad [\vec{\chi_p} \twoheadleftarrow \mathcal{A}y.\langle\ \vec{\tau_p}\ \rangle[\Psi]][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]\end{aligned}$$
$$\begin{aligned}&\{\!|\ \vec{\chi_p} \twoheadleftarrow \mathcal{A}y.\langle\ \vec{\tau_p}\ \rangle[\Psi][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]\ |\!\}_w = \\ &\qquad \{(\mathcal{A}y.[\![\ \tau_j\ ]\!]_w\{\!|\ \Psi\ |\!\}_w\{\!|\ \Theta\ |\!\}_w\{\bullet/z\})/\chi_j\} \\ &\qquad \longrightarrow^*_w \{(\mathcal{A}y.[\![\ \tau_j\ ]\!]_w\{\!|\ \Psi\ |\!\}_w)/\chi_j\}\{\!|\ \Theta\ |\!\}_w\{\bullet/z\} \\ &\qquad = \{\!|\ [\vec{\chi_p} \twoheadleftarrow \mathcal{A}y.\langle\ \vec{\tau_p}\ \rangle[\Psi]][\twoheadleftarrow \mathcal{A}z.\langle\ \rangle[\Theta]]\ |\!\}_w\end{aligned}$$

Duplication rules

1.

$$[\leftarrow @(t,\tau')] \longrightarrow_s [\leftarrow t][\leftarrow \tau']$$
$$[\leftarrow [\![\ @(t,\tau')\ ]\!]_w] \longrightarrow_w [\leftarrow [\![\ t\ ]\!]_w][\leftarrow [\![\ \tau'\ ]\!]_w]$$

2.

$$[\leftarrow \mathcal{A}x.t] \longrightarrow_s [\twoheadleftarrow \mathcal{A}x.\langle\ \rangle[\leftarrow t]]$$
$$[\leftarrow \mathcal{A}x.[\![\, t \,]\!]_w] \longrightarrow_w [\leftarrow [\![\, t \,]\!]_w]\{\bullet/x\}$$

3.

$$u^*[\twoheadleftarrow \mathcal{A}y\langle\ \rangle[\leftarrow y]] \longrightarrow_s u^*$$
$$[\![\, u^* \,]\!]_w[\leftarrow \bullet] \longrightarrow_w [\![\, u^* \,]\!]_w$$

□

5.2.1 Measuring weakening reductions

To show that weakening reductions in the weakening calculus are strongly normalizing, a measure can be constructed. The first component is the *size* of a term, the second component is the *depth* of a term which gathers multisets representing the paths to the sharings inside the term. Note that in the case of $T^*[\leftarrow U^*]$, we consider T^* to stay at the same depth as $T^*[\leftarrow U^*]$ because of the congruence rule. If $T^* = W^*[\phi][\psi] \sim W^*[\psi][\phi]$, the path to these closures should be of the same length.

Let $T^*, U^* \in \Lambda\mu S_w$ be terms or streams, let χ be a variable.

Definition 5.2.3. [Size] The size has signature:

$$\sigma : \underbrace{\Lambda\mu S_w}_{\text{term or stream } t^*} \rightarrow \underbrace{\mathbb{N}}_{\text{size}}$$

and is defined by structural induction on expressions:

- $\sigma(\chi) = \sigma(\bullet) = 1$.
- $\sigma(@(T, U^*)) = 1 + \sigma(T) + \sigma(U^*)$
- $\sigma(\mathcal{A}x.T) = 1 + \sigma(T)$.
- $\sigma(T^*[\leftarrow U^*]) \ = \sigma(T^*) + \sigma(U^*)$

To get the depth of an expression t^*, we measure $\partial(T^*, 1)$.

Definition 5.2.4. [Depth] The *depth* $\partial(T^*, n)$ measures a term T^* that is at depth n. It has signature:

$$\partial : \underbrace{\Lambda\mu S_w}_{\text{term or stream } t^*} \to \underbrace{\mathbb{N}}_{\text{input depth}} \to \underbrace{\mathcal{M}_f(\mathbb{N})}_{\text{output depths}}$$

We define it by induction on T^*:

- $\partial(\chi, n) = \emptyset$.
- $\partial(@(T, U^*), n) = \partial(T, n+1) + \partial(U^*, n+1)$.
- $\partial(\mathcal{A}x.T, n) = \partial(T, n+1)$.
- $\partial(T^*[\leftarrow U^*], n) \ = \partial(T^*, n) + \partial(U^*, n+1) + \{n\}$

Lemma 5.2.5 (Monotonicity of depth). For $T^* \in \Lambda\mu S_w, n \in \mathbb{N}$, $\partial(T^*, n) \leq \partial(T^*, n+1)$.

Proof. By induction:

- $\partial(\chi, n) = \emptyset = \partial(\chi, n+1)$.
- By induction hypothesis:

$$\begin{aligned}\partial(@(T, U^*), n) &= \partial(T, n+1) + \partial(U^*, n+1) \\ &\leq \partial(T, n+2) + \partial(U^*, n+2) = \partial(@(T, U^*), n+1)\end{aligned}$$

- By induction hypothesis:

$$\begin{aligned}\partial(\mathcal{A}x.T, n) &= \partial(T, n+1) \\ &\leq \partial(T, n+2) = \partial(\mathcal{A}x.T, n+1)\end{aligned}$$

- By induction hypothesis:

$$\begin{aligned}\partial(T^*[\leftarrow U^*], n) &= \partial(T^*, n) + \partial(U^*, n+1) + \{n\} \\ &\leq \partial(T^*, n+1) + \partial(U^*, n+2) + \{n+1\} \\ &= \partial(T^*[\leftarrow U^*], n+1)\end{aligned}$$

□

Lemma 5.2.6. For $U^* \in \Lambda\mu S_w$, if χ is a variable, $\sigma(U^*\{\bullet/\chi\}) = \sigma(U^*)$.

Proof. The proof follows from the fact that χ and $\bullet$ have the same size. □

Lemma 5.2.7. For $U^* \in \Lambda\mu S_w$, $\sigma(U^*[\leftarrow \mathcal{A}x.T]) > \sigma(U^*[\leftarrow T\{\bullet/x\}])$.

Proof. By Lemma 5.2.6, $\sigma(U^*\{\bullet/\chi\}) = \sigma(U^*)$. Then:

$$\begin{aligned}\sigma(U^*[\leftarrow \mathcal{A}x.T]) &= \sigma(U^*) + \sigma(\mathcal{A}x.T)\\ &= \sigma(U^*) + \sigma(T) + 1\\ &> \sigma(U^*) + \sigma(T\{\bullet/x\})\\ &= \sigma(U^*[\leftarrow T\{\bullet/x\}])\end{aligned}$$

□

5.2.2 SN for weakening reduction

We now show that weakening reductions are strongly normalizing, using our measure. Either the size strictly decreases, or it stays the same and the depth decreases.

Theorem 5.2.8. Weakening reductions are strongly normalizing in $\Lambda\mu S_w$.

Proof. • For lifting rules:

1. $\mathcal{A}x.(U[\leftarrow T^*]) \longrightarrow_w (\mathcal{A}x.U)[\leftarrow T^*]$ if $x \in FV(U)$

$$\begin{aligned}\sigma(\mathcal{A}x.(U[\leftarrow T^*])) &= 1 + \sigma(U[\leftarrow T^*])\\ &= 1 + \sigma(U) + \sigma(T^*)\\ \sigma((\mathcal{A}x.U)[\leftarrow T^*]) &= \sigma(\mathcal{A}x.U) + \sigma(T^*)\\ &= 1 + \sigma(U) + \sigma(T^*)\end{aligned}$$

$$\begin{aligned}\partial(\mathcal{A}x.(U[\leftarrow T^*]), n) &= \partial((U[\leftarrow T^*]), n+1)\\ &= \partial(U, n+1) + \partial(T^*, n+2) + \{n+1\}\\ \partial((\mathcal{A}x.U)[\leftarrow T^*], n) &= \partial((\mathcal{A}x.U), n) + \partial(T^*, n+1) + \{n\}\\ &= \partial(U, n+1) + \partial(T^*, n+1) + \{n\}\end{aligned}$$

We conclude with Lemma 5.2.5 (monotonicity), $\partial(T^*, n+2) \geq \partial(T^*, n+1)$.

2. $\begin{matrix}@(U[\leftarrow V^*], T^*) \\ @(U, T^*[\leftarrow V^*])\end{matrix} \longrightarrow_w @(U, T^*)[\leftarrow V^*]$

$$\sigma(@(U[\leftarrow V^*], T^*)) = 1 + \sigma(U) + \sigma(V^*) + \sigma(T^*)$$
$$\sigma(@(U, T^*)[\leftarrow V^*]) = 1 + \sigma(U) + \sigma(T^*) + \sigma(V^*)$$

$$\begin{aligned}\partial(@(T[\leftarrow V^*], U^*), n) &= \partial(T, n+1) + \partial(V^*, n+2) \\ &\quad + \{n+1\} + \partial(U^*, n+1) \\ \partial(@(T, U^*)[\leftarrow V^*], n) &= \partial(T, n+1) + \partial(U^*, n+1) \\ &\quad + \{n\} + \partial(V^*, n+1)\end{aligned}$$

3. $U^*[\leftarrow T^*[\leftarrow V^*]] \longrightarrow_w U^*[\leftarrow T^*][\leftarrow V^*]$

$$\sigma(U^*[\leftarrow T^*[\leftarrow V^*]]) = \sigma(U^*) + \sigma(T^*) + \sigma(V^*)$$
$$\sigma((U^*[\leftarrow T^*])[\leftarrow V^*]) = \sigma(U^*) + \sigma(T^*) + \sigma(V^*)$$

$$\begin{aligned}\partial(U^*[\leftarrow T^*[\leftarrow V^*]], n) &= \partial(U^*, n) + \partial(T^*, n+1) \\ &\quad + \{n\} + \partial(V^*, n+2) + \{n+1\} \\ \partial((U^*[\leftarrow T^*])[\leftarrow V^*], n) &= \partial(U^*, n) + \partial(T^*, n+1) \\ &\quad + \{n\} + \partial(V^*, n+1) + \{n\}\end{aligned}$$

- For compounding rules:

1. $U^*[\leftarrow T^*] \longrightarrow_w U^*$ if T^* is a subterm of U^*

$$\sigma(U^*[\leftarrow T^*]) = \sigma(U^*) + \sigma(T^*) > \sigma(U^*)$$

- For duplication rules:

1. $U^*[\leftarrow @(V^*, T^*)] \longrightarrow_w U^*[\leftarrow V^*][\leftarrow T^*]$

$$\begin{aligned}\sigma(U^*[\leftarrow @(V^*, T^*)]) &= \sigma(U^*) + \sigma(@(V^*, T^*)) \\ &= \sigma(U^*) + \sigma(V^*) + \sigma(T^*) + 1\end{aligned}$$

$$\begin{aligned}\sigma(U^*[\leftarrow V^*][\leftarrow T^*]) &= \sigma(U^*[\leftarrow V^*]) + \sigma(T^*)\\ &= \sigma(U^*) + \sigma(V^*) + \sigma(T^*)\end{aligned}$$

2. $U^*[\leftarrow \mathcal{A}x.T] \longrightarrow_w U^*[\leftarrow T\{\bullet/x\}]$

$$\begin{aligned}\sigma(U^*[\leftarrow \mathcal{A}x.T]) &= \sigma(U^*) + 1 + \sigma(T)\\ \sigma(U^*[\leftarrow T\{\bullet/x\}]) &= \sigma(U^*) + \sigma(T)\end{aligned}$$

   By Lemma 5.2.7.

3. $U^*[\leftarrow \bullet] \longrightarrow_w U^*$

$$\sigma(U^*[\leftarrow \bullet]) = \sigma(U^*) + 1 > \sigma(U^*)$$

□

5.3 Weight

We now consider our second measure for sharing reductions in the atomic calculus. The idea behind measuring the *weight* of an expression is to quantify the remaining duplications, which are performed with sharing reductions. To give an intuition, take the term $(x_1)x_2 \ldots x_p[x_1, \ldots, x_p \leftarrow w]$. The sharing $[x_1, \ldots, x_p \leftarrow w]$ represents p instances of w, and we obtain p copies of w after unfolding the sharing. The weight p can thus be assigned inductively to w. For instance, if there are no sharings in w, any variable of w would also have weight p. In a sharing $[\chi_1, \ldots, \chi_p \leftarrow w^*]$, the weight of the term w^* corresponds to the sum of the weights of the χ_i. For weakenings, suppose $t^*[\leftarrow u^*]$ has weight n. Since we would still like to measure weights inside u^*, intuitively we can either consider that u^* has weight 1 (u^* can eventually be lifted, so it can be counted once), or that t^*, u^* are subterms of $t^*[\leftarrow u^*]$ that keep the same weight $n > 0$. Our definition follows the second choice. We thus also directly measure how many copies there are of a term when translating into $\Lambda\mu S_w$. The weight measure consists of collecting all weights of all subexpressions in a multiset. For the proof of strong normalization of $\longrightarrow_s$, we will show that the measure reduces after a reduction step.

To summarize, in order to compute the weights of a subexpression $t^*[\vec{\chi_p} \leftarrow u^*]$ of weight n, we:

1. Compute the weights of the χ_i in t^*
2. For u^*, the weight is the sum of the weights of χ_i

We thus construct a function ω:

- With inputs:
  - the expression $t^*[\vec{\chi_p} \leftarrow u^*]$
  - its initial weight n
- With outputs:
  - the multiset of weights of all subexpressions
  - a function assigning a weight to each free variable

Because we want to measure the copies of a subexpression when translated to $\Lambda\mu S_w$, we don't measure the weight of variables bound by sharings. More explicitly, for a term $t^*[\vec{\chi_p} \leftarrow u^*] \in \Lambda\mu S_a$, in its translation $T^*\{U^*/\chi_1\}\dots\{U^*/\chi_p\} \in \Lambda\mu S_w$ the variables χ_i are substituted for. However, we want to measure variables bound by abstractions, so free variables are weighed in the second output, and are weighed in the first (main) measure when they become abstracted over. When measuring $\mathcal{A}x.t$, we add the weight of the whole term, and the weight of x in t to the first output measure.

Example 5.3.1. For instance, consider the weight $\omega(u, 1)$ for

$$u = \lambda x.(x_1)x_2[x_1, x_2 \leftarrow x] = \lambda x.t$$

We have

$$\omega(x_i, 1) = (\emptyset, x_i \mapsto 1)$$

Therefore we consider:

$$\omega(x, 2) = (\emptyset, x \mapsto 2)$$

summing the weights of x_1 and x_2. Counting the application constructor, we have:

$$\omega(t, 1) = (\{1\}, x \mapsto 2)$$

Then, counting the abstraction constructor and transferring the weight of x to the first measure, we have:

$$\omega(u, 1) = (\{1^2, 2\}, \emptyset)$$

Finally for distributors, we consider that the introduction rule

$$[\vec{x_p} \leftarrow \mathcal{A}y.t] \longrightarrow_s [\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t]]$$

duplicates abstractions $\mathcal{A}y$, without copying y yet. This variable is copied when eliminating the distributor

$$[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\vec{y_p} \leftarrow y]] \longrightarrow_s \{\mathcal{A}y_i.t_i[\vec{y_i^{l_i}} \leftarrow y_i]/x_i\}$$

This can be illustrated with the graphical representation below. An expression t^* can be represented as a tree where the nodes are annotated with a weight (denoted by $n, n', \dots$) intuitively corresponding to the number of "copies" of subexpressions of t^*. Recall that t^* denotes a term, a stream or a tuple.

Let $\tau \in \mathbb{T} \cup \mathbb{S}$ be a term or a stream, let χ, χ_i be variables. In the picture below, the overlined blue indices represent the weights of each connective, as measured by the weight function and collected in the first output m, whereas the underlined red indices give the input weights, but also the output weights to free variables that do not contribute to the measure. The weight counts the number of copies of free variables χ, variables bound by $\mathcal{A}$, and function symbols $\mathcal{A}, @$.

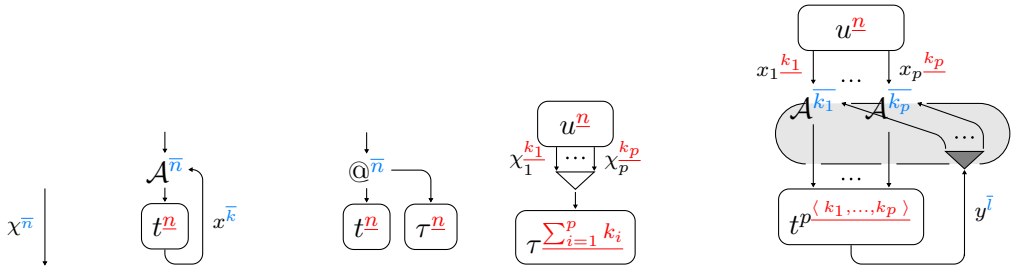


Figure 5-1: Graphical representation of weights

To prove the preservation of strong normalization, the *weight* is the first measure, composed of a finite multiset m and of a function f, where n represents the weight of t^*, and p the arity of t^*. For the measure, we solely take into account the first output m, while f is only used to get the input to recursive calls of the weight function on shared terms. The signature of the weight function is as follows:

$$\begin{aligned}
\omega : & \underbrace{\mathcal{T}^*}_{\text{term, stream, tuple } t^*} \to \underbrace{\mathbb{N}^p}_{\text{arity of } t^*} \to \underbrace{(\mathbb{N} \to \mathbb{N})}_{\text{main measure } m} \times \underbrace{(\mathcal{V} \to \mathbb{N})}_{\text{measure of free variables } f} \\
= & \underbrace{\mathcal{T}^*}_{\text{term, stream, tuple } t^*} \to \underbrace{\mathbb{N}^p}_{\text{arity of } t^*} \to \underbrace{\mathcal{M}_f(\mathbb{N})}_{\text{main measure } m} \times \underbrace{\mathcal{M}_f(\mathcal{V})}_{\text{measure of free variables } f}
\end{aligned}$$

Remark 5.3.2. The set $\mathcal{V}$ of free variables is such that $\mathcal{V} = \mathcal{V}_\lambda \sqcup \mathcal{V}_\mu$, where $\mathcal{V}_\lambda$ is a set of free λ-variables, and $\mathcal{V}_\mu$ is a set of free μ-variables.

For any term t, we measure $\omega(t, 1)$. A 0-tuple u^0 appears in weakened distributors $[\nleftarrow \mathcal{A}y.u^0]$, and its behavior is similar to that of a term or stream τ appearing in a weakening $[\leftarrow \tau]$. For this reason if t^* is a term, a stream, or a 0-tuple, we consider that it has arity 1, otherwise a tuple t^p has arity $p \geq 2$. For a q-term t^q we will instead need $\omega(t^q, \vec{n_q})$ to separately count the weights of each projection on the i-th element of t^q.

We now define $\omega(t^*, \vec{n_q})$ for a term of arity q and of weight $\vec{n_q}$. Figure 5-1 describes the different cases.

For tuples, we gather the weights of each coordinate of the tuple. Since all occurrences are linear, it is possible to simply sum all functions. For applications and abstractions, the constructors $\mathcal{A}, @$ are counted in the weight, then the function is applied recursively to subterms. For sharings, we sum the weight of sharing variables χ_i, which corresponds to the weight of the shared expression τ. Since the variables χ_i become bound by the sharing, their weight is removed. For distributors, the weights to count are the abstraction constructor $\mathcal{A}$, the tuples, and the variable y that becomes abstracted over. The abstraction $\mathcal{A}y.u^p$ is distributed to the variables x_i (so its weight depends on each x_i), then the function is applied inductively on subterms.

Definition 5.3.3. [Weight] The weight of $\Lambda\mu S_a$-expressions is defined as follows:

- $\omega(\chi, n) = (\emptyset, \{\chi^n\})$
- $\omega(\langle\ \rangle, n) = (\emptyset, \emptyset)$.
- $\omega(\langle\, t_1, \ldots, t_p\, \rangle, n_1, \ldots, n_p) = (\sum_{i=1}^{p} m_i, \sum_{i=1}^{p} f_i)$
  for $1 \leq i \leq p$, where $\omega(t_i, n_i) = (m_i, f_i)$.
- $\omega(@(t, \tau), n) = (m_t + m_\tau + \{n\}, f_t + f_\tau)$
  where $\omega(t, n) = (m_t, f_t)$, and $\omega(\tau, n) = (m_\tau, f_\tau)$.
- $\omega(\mathcal{A}x.t, n) = (m_t + \{n\} + \{f_t(x)\}, f_t \backslash x)$
  where $\omega(t, n) = (m_t, f_t)$.
- $\omega(t^*[\leftarrow \tau], n_1, \ldots, n_p) = (m_{t^*} + \sum_{i=1}^{p} m_i, f_{t^*} + \sum_{i=1}^{p} f_i)$
  where $\omega(t^*, n_1, \ldots, n_p) = (m_{t^*}, f_{t^*})$, and $\omega(\tau, n_i) = (m_i, f_i)$.
- $\omega(t^*[\chi_1, \ldots, \chi_p \leftarrow \tau], n_1, \ldots, n_q) \ = (m_{t^*} + m_\tau, (f_{t^*} \backslash \chi_1, \ldots, \chi_p) + f_\tau)$

where $\omega(t^*, n_1, \ldots, n_q) = (m_{t^*}, f_{t^*})$, and $\omega(\tau, r) = (m_\tau, f_\tau)$,

where $r = \sum_{i=1}^{q} f_{t^*}(\chi_i)$.

- $\omega(t^*[\twoheadleftarrow \mathcal{A}y.u^0], n_1, \ldots, n_q) = (m, f)$ where

$$\begin{aligned} m &= m_{t^*} + \sum_{i=1}^{q} m_i + \{n_i \mid 1 \leq i \leq q\} + \{f_i(y) \mid 1 \leq i \leq q\} \\ f &= f_{t^*} + ((\sum_{i=1}^{q} f_i) \backslash y) \end{aligned}$$

$$\begin{aligned} \omega(t^*, n_1, \ldots, n_q) &= (m_{t^*}, f_{t^*}) \\ \omega(u^0, n_i) &= (m_i, f_i) \end{aligned}$$

- $\omega(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p], n_1, \ldots, n_q) = (m, f)$ where

$$\begin{aligned} m &= m_{t^*} + m_{u^p} + \{f_{t^*}(x_i) \mid 1 \leq i \leq p\} + \{f_{u^p}(y)\} \\ f &= (f_{t^*} \backslash x_1, \ldots, x_p) + (f_{u^p} \backslash y) \end{aligned}$$

$$\begin{aligned} \omega(t^*, n_1, \ldots, n_q) &= (m_{t^*}, f_{t^*}) \\ \omega(u^p, f_{t^*}(x_1), \ldots, f_{t^*}(x_p)) &= (m_{u^p}, f_{u^p}) \end{aligned}$$

The lemma below shows that weights behave well with substitution, i.e. an expression τ replacing χ in t^* replaces the weight of χ with that of τ.

Lemma 5.3.4. Let $t^* \in \Lambda\mu S_a$ be of arity q. Then $\omega(t^*\{\tau/\chi\}, \vec{n_q}) = (m_{t^*} + m_\tau, (f_{t^*} \backslash \chi) + f_\tau)$ where $\omega(t^*, \vec{n_q}) = (m_{t^*}, f_{t^*})$ and $\omega(\tau, f_{t^*}(\chi)) = (m_\tau, f_\tau)$.

Proof. Note that since we work in $\Lambda\mu S_a$, the variable χ must appear exactly once in t^*, and is not bound.

1. $\chi\{\tau/\chi\} = \tau$.

   Therefore:

   $\omega(\tau, n) = (m_\tau, f_\tau) = (m_\tau, (\{\chi^n\} \backslash \chi) + f_\tau)$,

   where $\omega(\chi, n) = (\emptyset, \{\chi^n\})$.

2. $\langle\, t_1, \ldots, t_q \,\rangle\{\tau/\chi\} = \langle\, t_1, \ldots, t_i\{\tau/\chi\}, \ldots, t_q \,\rangle$ if χ appears in t_i.

   By induction hypothesis:

   $\omega(t_i\{\tau/\chi\}, n_i) = (m_i + m_\tau, (f_i \backslash \chi) + f_\tau)$.

   Then,

$$\begin{aligned}&\omega(\langle\, t_1, \ldots, t_i\{\tau/\chi\}, \ldots, t_q \,\rangle, n_1, \ldots, n_q)\\ &= (\underbrace{\sum_{k=1}^{q} m_k}_{\langle\, \vec{t_q} \,\rangle} + m_\tau, \underbrace{\sum_{k=1, k\neq i}^{q} f_k + f_i \backslash \chi}_{\langle\, \vec{t_q} \,\rangle} + f_\tau)\end{aligned}$$

   where $\omega(t_k, n_k) = (m_k, f_k)$.

3. $@(t, \tau')\{\tau/\chi\} = @(t\{\tau/\chi\}, \tau')$

   (respectively $@(t, \tau'\{\tau/\chi\})$ if χ appears in t (respectively in τ').

   - Suppose χ appears in t.

     By induction hypothesis:

     $\omega(t\{\tau/\chi\}, n) = (m_t + m_\tau, f_t \backslash \chi + f_\tau)$.

     Then,

$$\begin{aligned}&\omega(@(t\{\tau/\chi\}, \tau'), n)\\ &= (m_t + m_\tau + \{n\} + m_{\tau'}, f_t \backslash \chi + f_\tau + f_{\tau'})\\ &= (\underbrace{m_t + m_{\tau'} + \{n\}}_{@(t,\tau')} + m_\tau, \underbrace{f_{\tau'} + f_t}_{@(t,\tau')} \backslash \chi + f_\tau)\end{aligned}$$

     where $\omega(t, n) = (m_t, f_t)$, and $\omega(\tau', n) = (m'_\tau, f'_\tau)$.

   - Suppose χ appears in τ'.

     By induction hypothesis:

     $\omega(\tau'\{\tau/\chi\}, n) = (m_{\tau'} + m_\tau, (f_{\tau'} \backslash \chi) + f_\tau)$.

     Then,

$$\begin{aligned}\omega(@(t, \tau'\{\tau/\chi\}), n) &= (m_t + \{n\} + m_{\tau'} + m_\tau, f_t + f_{\tau'} \backslash \chi + f_\tau)\\ &= (\underbrace{m_t + m_{\tau'} + \{n\}}_{@(t,\tau')} + m_\tau, \underbrace{f_t + f_{\tau'}}_{@(t,\tau')} \backslash \chi + f_\tau)\end{aligned}$$

     where $\omega(t, n) = (m_t, f_t)$, and $\omega(\tau', n) = (m'_\tau, f'_\tau)$.

4. $(\mathcal{A}x.t)\{\tau/\chi\} = \mathcal{A}x.(t\{\tau/\chi\})$.

   Therefore, by induction hypothesis:

   $\omega(t\{\tau/\chi\}, n) = (m_t + m_\tau, (f_t \backslash \chi) + f_\tau)$.

   Then,

$$\begin{aligned}\omega(\mathcal{A}x.(t\{\tau/\chi\}), n) &= (m_t + m_\tau + \{n\} + \{f_t(x)\}, (f_t\backslash\chi, x) + f_\tau) \\ &= (\underbrace{m_t + \{n\} + \{f_t(x)\}}_{\mathcal{A}x.t} + m_\tau, \underbrace{(f_t\backslash x}_{\mathcal{A}x.t}, \chi) + f_\tau)\end{aligned}$$

   where $\omega(t, n) = (m_t, f_t)$ and $\omega(\tau', n) = (m_{\tau'}, f_{\tau'})$.

5. $(t^*[\leftarrow \tau'])\{\tau/\chi\} = (t^*\{\tau/\chi\})[\leftarrow \tau']$ (respectively $t^*[\leftarrow \tau'\{\tau/\chi\}]$) if χ appears in t^* (respectively τ').

   - Suppose that χ appears in t^*.

     By induction hypothesis:

     $\omega(t^*\{\tau/\chi\}, \vec{n_q}) = (m_{t^*} + m_\tau, (f_{t^*}\backslash\chi) + f_\tau)$.

     Then,

$$\begin{aligned}&\omega((t^*[\leftarrow \tau'])\{\tau/\chi\}, n_1, \ldots, n_q) \\ &= (m_{t^*} + m_\tau + \sum_{i=1}^{q} m_i, f_{t^*}\backslash\chi + f_\tau + \sum_{i=1}^{q} f_i) \\ &= (\underbrace{m_{t^*} + \sum_{i=1}^{q} m_i}_{t^*\{\tau/\chi\}} + m_\tau, \underbrace{\sum_{i=1}^{q} f_i + f_{t^*}\backslash\chi}_{t^*\{\tau/\chi\}} + f_\tau)\end{aligned}$$

     where $\omega(t^*, n_1, \ldots, n_q) = (m_{t^*}, f_{t^*})$, and $\omega(\tau', n_i) = (m_i, f_i)$.

   - Suppose that χ appears in τ'.

     By induction hypothesis: $\omega(\tau'\{\tau/\chi\}, n_i) = (m_i + m_{\tau'}, (f_i\backslash\chi) + f_{\tau'})$.

Then,

$$\begin{aligned}
&\omega((t^*[\leftarrow \tau'])\{\tau/\chi\}, n_1, \ldots, n_q) \\
&= (m_{t^*} + \sum_{i=1}^{q} m_i + m_\tau, f_{t^*} + f_\tau + \sum_{i=1}^{q} f_i \backslash \chi) \\
&= (\underbrace{m_\tau + \sum_{i=1}^{q} m_i}_{\tau'\{\tau/\chi\}} + m_{t^*}, \underbrace{f_\tau + \sum_{i=1}^{q} f_i \backslash \chi}_{\tau'\{\tau/\chi\}} + f_{t^*})
\end{aligned}$$

where $\omega(t^*, n_1, \ldots, n_q) = (m_{t^*}, f_{t^*})$, and $\omega(\tau', n_i) = (m_i, f_i)$.

6. $(t^*[\chi_1, \ldots, \chi_p \leftarrow \tau'])\{\tau/\chi\} = (t^*\{\tau/\chi\}[\chi_1, \ldots, \chi_p \leftarrow \tau'])$
   (respectively $t^*[\chi_1, \ldots, \chi_p \leftarrow \tau'\{\tau/\chi\}]$) if χ appears in t^* (respectively τ').

   - Suppose that χ appears in t^*.
     By induction hypothesis:
     $\omega(t^*\{\tau/\chi\}, \vec{n_q}) = (m_{t^*} + m_\tau, (f_{t^*} \backslash \chi) + f_\tau)$.
     Then,

$$\begin{aligned}
&\omega(t^*\{\tau/\chi\}[\chi_1, \ldots, \chi_p \leftarrow \tau'], n_1, \ldots, n_q) \\
&= (m_{t^*} + m_\tau + m_{\tau'}, (f_{t^*} \backslash \chi_1, \ldots, \chi_p, \chi) + f_\tau + f_{\tau'}) \\
&= (\underbrace{m_{t^*} + m_{\tau'}}_{t^*[\chi_1, \ldots, \chi_p \leftarrow \tau']} + m_\tau, \underbrace{f_{\tau'} + f_{t^*} \backslash \chi_1, \ldots, \chi_p}_{t^*[\chi_1, \ldots, \chi_p \leftarrow \tau']}, \chi + f_\tau)
\end{aligned}$$

     where
     $\omega(t^*, n_1, \ldots, n_q) = (m_{t^*}, f_{t^*})$, and $\omega(\tau', r) = (m_{\tau'}, f_{\tau'})$,
     where $r = \sum_{i=1}^{p} f_{t^*}(\chi_i)$.

   - Suppose χ appears in τ'.
     By induction hypothesis:
     $\omega(\tau'\{\tau/\chi\}, \vec{n_q}) = (m_{\tau'} + m_\tau, (f_{\tau'} \backslash \chi) + f_\tau)$.
     Then,

$$\begin{aligned}
&\omega(t^*[\chi_1, \ldots, \chi_p \leftarrow \tau'\{\tau/\chi\}], n_1, \ldots, n_q) \\
&= (\underbrace{m_{t^*} + m_{\tau'}}_{t^*[\chi_1, \ldots, \chi_p \leftarrow \tau']} + m_\tau, \underbrace{(f_{t^*} \backslash \chi_1, \ldots, \chi_p) + f_{\tau'}}_{t^*[\chi_1, \ldots, \chi_p \leftarrow \tau']} \backslash \chi + f_\tau)
\end{aligned}$$

where

$\omega(t^*, n_1, \ldots, n_q) = (m_{t^*}, f_{t^*})$, and

$\omega(\tau', r) = (m_{\tau'}, f_{\tau'})$,

where $r = \sum_{i=1}^{p} f_{t^*}(\chi_i)$.

7. $(t^*[\twoheadleftarrow \mathcal{A}y.u^0])\{\tau/\chi\} = (t^*\{\tau/\chi\}[\twoheadleftarrow \mathcal{A}y.u^0])$ (respectively $t^*[\twoheadleftarrow \mathcal{A}y.u^0\{\tau/\chi\}]$) if χ appears in t^* (respectively u^0).

   - Suppose that χ appears in t^*.

     By induction hypothesis:

     $\omega(t^*\{\tau/\chi\}, \vec{n_q}) = (m_{t^*} + m_\tau, (f_{t^*} \backslash \chi) + f_\tau)$.

     Then,

$$
\begin{aligned}
&\omega(t^*\{\tau/\chi\}[\twoheadleftarrow \mathcal{A}y.u^0], n_1, \ldots, n_q) \\
&= (m_{t^*} + m_\tau + \sum_{i=1}^{q} m_i + \{n_i \mid 1 \le i \le q\} + \{f_i(y) \mid 1 \le i \le q\}, \\
&f_{t^*} \backslash \chi + f_\tau + ((\sum_{i=1}^{q} f_i) \backslash y)) \\
&= \underbrace{(m_{t^*} + \sum_{i=1}^{q} m_i + \{n_i \mid 1 \le i \le q\} + \{f_i(y) \mid 1 \le i \le q\}}_{t^*[\twoheadleftarrow \mathcal{A}y.u^0]} + m_\tau, \\
&\underbrace{((\sum_{i=1}^{q} f_i) \backslash y) + f_{t^*}}_{t^*[\twoheadleftarrow \mathcal{A}y.u^0]} \backslash \chi + f_\tau)
\end{aligned}
$$

     where

$$
\begin{aligned}
\omega(t^*, n_1, \ldots, n_q) &= (m_{t^*}, f_{t^*}) \\
\omega(u^0, n_i) &= (m_i, f_i)
\end{aligned}
$$

   - Suppose χ appears in u^0.

     By induction hypothesis:

     $\omega(u^0\{\tau/\chi\}, \vec{n_q}) = (m_{u^0} + m_\tau, (f_{u^0} \backslash \chi) + f_\tau)$.

Then,

$$\begin{aligned}
&\omega(t^*[\twoheadleftarrow \mathcal{A}y.u^0\{\tau/\chi\}], n_1, \ldots, n_q) \\
&= \underbrace{(m_{t^*} + \sum_{i=1}^{q} m_i + \{n_i \mid 1 \leq i \leq q\} + \{f_i(y) \mid 1 \leq i \leq q\}}_{t^*[\twoheadleftarrow \mathcal{A}y.u^0]} + m_\tau, \\
&\underbrace{f_{t^*} + ((\sum_{i=1}^{q} f_i)\backslash y, \chi) + f_\tau)}_{t^*[\twoheadleftarrow \mathcal{A}y.u^0]}
\end{aligned}$$

where

$$\begin{aligned}
\omega(t^*, n_1, \ldots, n_q) &= (m_{t^*}, f_{t^*}) \\
\omega(u^0, n_i) &= (m_i, f_i)
\end{aligned}$$

8. $(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p])\{\tau/\chi\} = (t^*\{\tau/\chi\}[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p])$ (respectively $(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p\{\tau/\chi\}])$) if χ appears in t^* (respectively u^p).

   - Suppose that χ appears in t^*.

     By induction hypothesis:

     $\omega(t^*\{\tau/\chi\}, \vec{n_q}) = (m_{t^*} + m_\tau, (f_{t^*}\backslash\chi) + f_\tau)$.

     Then,

$$\begin{aligned}
&\omega(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p], n_1, \ldots, n_q) \\
&= (m_{t^*} + m_\tau + m_{u^p} + \{f_{t^*}(x_i) \mid 1 \leq i \leq p\} + \{f_{u^p}(y)\}, \\
&(f_{t^*}\backslash x_1, \ldots, x_p, \chi) + f_\tau + (f_{u^p}\backslash y)) \\
&= \underbrace{(m_{t^*} + m_{u^p} + \{f_{t^*}(x_i) \mid 1 \leq i \leq p\} + \{f_{u^p}(y)\}}_{t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p]} + m_\tau, \\
&\underbrace{(f_{u^p}\backslash y) + f_{t^*}\backslash x_1, \ldots, x_p}_{t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p]}, \chi + f_\tau)
\end{aligned}$$

     where

$$\begin{aligned}
\omega(t^*, n_1, \ldots, n_q) &= (m_{t^*}, f_{t^*}) \\
\omega(u^p, f_{t^*}(x_1), \ldots, f_{t^*}(x_p)) &= (m_{u^p}, f_{u^p})
\end{aligned}$$

   - Suppose χ appears in u^p.

By induction hypothesis:
$\omega(u^p\{\tau/\chi\}, \vec{n_q}) = (m_{u^p} + m_\tau, (f_{u^p} \backslash \chi) + f_\tau)$.
Then,

$$\begin{aligned}
&\omega(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p], n_1, \ldots, n_q) \\
&= (m_{t^*} + m_{u^p} + m_\tau + \{f_{t^*}(x_i) \mid 1 \leq i \leq p\} + \{f_{u^p}(y)\}, \\
&(f_{t^*} \backslash x_1, \ldots, x_p) + (f_{u^p} \backslash y, \chi) + f_\tau) \\
&= (\underbrace{m_{t^*} + m_{u^p} + \{f_{t^*}(x_i) \mid 1 \leq i \leq p\} + \{f_{u^p}(y)\}}_{t^*[x_1,\ldots,x_p \twoheadleftarrow \mathcal{A}y.u^p]} + m_\tau, \\
&\underbrace{(f_{t^*} \backslash x_1, \ldots, x_p) + (f_{u^p} \backslash y}_{t^*[x_1,\ldots,x_p \twoheadleftarrow \mathcal{A}y.u^p]}, \chi) + f_\tau)
\end{aligned}$$

where

$$\begin{aligned}
\omega(t^*, n_1, \ldots, n_q) &= (m_{t^*}, f_{t^*}) \\
\omega(u^p, f_{t^*}(x_1), \ldots, f_{t^*}(x_p)) &= (m_{u^p}, f_{u^p})
\end{aligned}$$

□

By definition of $\omega(t^*[\chi \leftarrow \tau], \vec{n_q})$, we get the following corollary:

Corollary 5.3.5. Let $t^* \in \Lambda\mu S_a$. Then, $\omega(t^*\{\tau/\chi\}, \vec{n_q}) = \omega(t^*[\chi \leftarrow \tau], \vec{n_q})$.

From the following lemma we can deduce the monotonicity of ω, i.e. $\omega(t^*, \vec{n_k}) \leq \omega(t^*, \vec{m_k})$ if $n_i \leq m_i$ for $1 \leq i \leq k$.

Lemma 5.3.6. For $t^* \in \Lambda\mu S_a$, $\omega(t^*, \vec{m_k}, n, \vec{p_l}) \leq \omega(t^*, \vec{m_k}, n+1, \vec{p_l})$.

Also, let $\chi \in FV(t^*)$, $\omega(t^*, \vec{m_k}, n, \vec{p_l}) = (m_{t^*}, f_{t^*})$, and $\omega(t^*, \vec{m_k}, n+1, \vec{p_l}) = (m'_{t^*}, f'_{t^*})$. Then $f_{t^*}(\chi) \leq f'_{t^*}(\chi)$.

Proof. By induction on t^*.

- For a variable χ:

$$\begin{aligned}
\omega(\chi, n) &= (\emptyset, \{\chi^n\}) \\
\omega(\chi, n+1) &= (\emptyset, \{\chi^{n+1}\})
\end{aligned}$$

- For an empty tuple:

$$\begin{aligned}\omega(\langle\ \rangle, n) &= (\emptyset, \emptyset) \\ &= \omega(\langle\ \rangle, n+1)\end{aligned}$$

- For tuples:

$$\omega(\langle\, t_1, \ldots, t_p\, \rangle, n_1, \ldots, n_p) = (\sum_{i=1}^{p} m_i, \sum_{i=1}^{p} f_i)$$

  Then:

$$\begin{aligned}&\omega(\langle\, t_1, \ldots, t_p\, \rangle, n_1, \ldots, n_s + 1, \ldots, n_p) \\ &= (\sum_{i=1, i\neq s}^{p} m_i + m'_s, \sum_{i=1, i\neq s}^{p} f_i + f'_s)\end{aligned}$$

  for $1 \leq i \leq p$, where

$$\begin{aligned}\omega(t_i, n_i) &= (m_i, f_i) \\ \omega(t_s, n_s + 1) &= (m'_s, f'_s)\end{aligned}$$

  We apply our induction hypothesis $\omega(t_s, n_s) \leq \omega(t_s, n_s + 1)$.

- For applications:

$$\omega(@(t, \tau), n) = (m_t + m_\tau + \{n\}, f_t + f_\tau)$$

  Then:

$$\omega(@(t, \tau), n + 1) = (m'_t + m'_\tau + \{n + 1\}, f'_t + f'_\tau)$$

  where

$$\begin{aligned}\omega(t, n) &= (m_t, f_t) \\ \omega(\tau, n) &= (m_\tau, f_\tau) \\ \omega(t, n + 1) &= (m'_t, f'_t) \\ \omega(\tau, n + 1) &= (m'_\tau, f'_\tau)\end{aligned}$$

  We apply our induction hypothesis, therefore $\omega(t, n) \leq \omega(t, n + 1)$ and $\omega(\tau, n) \leq$

$\omega(\tau, n+1)$.

- For abstractions:

$$\omega(\mathcal{A}x.t, n) = (m_t + \{n\} + \{f_t(x)\}, f_t \backslash x)$$

Then:

$$\omega(\mathcal{A}x.t, n+1) = (m'_t + \{n+1\} + \{f'_t(x)\}, f'_t \backslash x)$$

where

$$\begin{aligned} \omega(t, n) &= (m_t, f_t) \\ \omega(t, n+1) &= (m'_t, f'_t) \end{aligned}$$

We apply our induction hypothesis $\omega(t, n) \leq \omega(t, n+1)$.

- For weakened sharings:

$$\omega(t^*[\leftarrow \tau], n_1, \ldots, n_p) = (m_{t^*} + \sum_{i=1}^{p} m_i, f_{t^*} + \sum_{i=1}^{p} f_i)$$

Then:

$$\begin{aligned} &\omega(t^*[\leftarrow \tau], n_1, \ldots, n_s + 1, \ldots, n_p) \\ &= (m'_{t^*} + \sum_{i=1, i\neq s}^{p} m_i + m'_s, f'_{t^*} + \sum_{i=1, i\neq s}^{p} f_i + f'_s) \end{aligned}$$

where

$$\begin{aligned} \omega(t^*, n_1, \ldots, n_p) &= (m_{t^*}, f_{t^*}) \\ \omega(\tau, n_i) &= (m_i, f_i) \\ \omega(t^*, n_1, \ldots, n_s + 1, \ldots, n_p) &= (m'_{t^*}, f'_{t^*}) \\ \omega(\tau, n_s + 1) &= (m'_s, f'_s) \end{aligned}$$

We apply our induction hypothesis on t^*, and $\omega(\tau, n_s) \leq \omega(\tau, n_s + 1)$.

- For sharings:

$$\omega(t^*[\chi_1, \ldots, \chi_p \leftarrow \tau], n_1, \ldots, n_q) = (m_{t^*} + m_\tau, (f_{t^*} \backslash \chi_1, \ldots, \chi_p) + f_\tau)$$

Then:

$$\begin{aligned}&\omega(t^*[\chi_1,\ldots,\chi_p \leftarrow \tau], n_1,\ldots,n_s+1,\ldots,n_q)\\&= (m'_{t^*} + m'_\tau, (f'_{t^*}\backslash\chi_1,\ldots,\chi_p) + f'_\tau)\end{aligned}$$

where

$$\begin{aligned}\omega(t^*, n_1,\ldots,n_q) &= (m_{t^*}, f_{t^*})\\ \omega(\tau, r) &= (m_\tau, f_\tau)\\ r &= \sum_{i=1}^{q} f_{t^*}(\chi_i)\\ \omega(t^*, n_1,\ldots,n_s+1,\ldots,n_q) &= (m'_{t^*}, f'_{t^*})\\ \omega(\tau, r') &= (m'_\tau, f'_\tau)\\ r' &= \sum_{i=1}^{q} f'_{t^*}(\chi_i)\end{aligned}$$

By induction on t^*, and since $f_{t^*}(\chi_i) \leq f'_{t^*}(\chi_i)$.

- For weakened distributors:

$$\omega(t^*[\twoheadleftarrow \mathcal{A}y.u^0], n_1,\ldots,n_q) = (m, f)$$

where

$$\begin{aligned}m &= m_{t^*} + \sum_{i=1}^{q} m_i + \{n_i \mid 1 \leq i \leq q\} + \{f_i(y) \mid 1 \leq i \leq q\}\\ f &= f_{t^*} + ((\sum_{i=1}^{q} f_i)\backslash y)\end{aligned}$$

Then:

$$\omega(t^*[\twoheadleftarrow \mathcal{A}y.u^0], n_1,\ldots,n_s+1,\ldots,n_q) = (m', f')$$

where

$$\begin{aligned} m' &= m'_{t^*} + \sum_{i=1, i\neq s}^{q} m_i + m'_s + \{n_i \mid 1 \leq i \leq q, i \neq s\} + \\ &\quad \{n_s + 1\} + \{f_i(y) \mid 1 \leq i \leq q, i \neq s\} + \{f'_s(y)\} \\ f' &= f'_{t^*} + (((\sum_{i=1, i\neq s}^{q} f_i) + f'_s) \backslash y) \end{aligned}$$

$$\begin{aligned} \omega(t^*, n_1, \ldots, n_q) &= (m_{t^*}, f_{t^*}) \\ \omega(u^0, n_i) &= (m_i, f_i) \\ \omega(t^*, n_1, \ldots, n_s + 1, \ldots, n_q) &= (m'_{t^*}, f'_{t^*}) \\ \omega(u^0, n_s + 1) &= (m'_s, f'_s) \end{aligned}$$

By induction on t^*, and $\omega(u^0, n_s) \leq \omega(u^0, n_s + 1)$, and $f_s(y) \leq f'_s(y)$.

- For distributors:

$$\omega(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p], n_1, \ldots, n_q) = (m, f)$$

where

$$\begin{aligned} m &= m_{t^*} + m_{u^p} + \{f_{t^*}(x_i) \mid 1 \leq i \leq p\} + \{f_{u^p}(y)\} \\ f &= (f_{t^*} \backslash x_1, \ldots, x_p) + (f_{u^p} \backslash y) \end{aligned}$$

Then:

$$\omega(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p], n_1, \ldots, n_s + 1, \ldots, n_q) = (m, f)$$

where

$$\begin{aligned} m' &= m'_{t^*} + m'_{u^p} + \{f'_{t^*}(x_i) \mid 1 \leq i \leq p\} + \{f'_{u^p}(y)\} \\ f' &= (f'_{t^*} \backslash x_1, \ldots, x_p) + (f'_{u^p} \backslash y) \end{aligned}$$

$$\omega(t^*, n_1, \ldots, n_q) = (m_{t^*}, f_{t^*})$$
$$\omega(u^p, f_{t^*}(x_1), \ldots, f_{t^*}(x_p)) = (m_{u^p}, f_{u^p})$$
$$\omega(t^*, n_1, \ldots, n_s + 1, \ldots, n_q) = (m'_{t^*}, f'_{t^*})$$
$$\omega(u^p, f'_{t^*}(x_1), \ldots, f'_{t^*}(x_p)) = (m'_{u^p}, f'_{u^p})$$

By induction on t^*, u^p, and $f_{t^*}(x_i) \leq f'_{t^*}(x_i)$ and $f_{u^p}(y) \leq f'_{u^p}(y)$.

□

5.4 Depth

We now introduce another part of the normalization measure, the *depth*, which is the length of the paths from each closure to the root of the term. It extends depth from weakening calculus (see Definition 5.2.4). In our definition we omit all cases redundant with the weakening calculus. Note that depth is the same when considering weakened sharings and non-weakened sharings. In the informal representation below, we consider that the terms illustrated are at depth n. The output of the depth function is shown in overlined $\overline{\text{blue}}$, and the parameters passed recursively are shown in underlined $\underline{\text{red}}$. Variables χ do not involve closures, so their depth output is 0. For expressions $\mathcal{A}x.t, @(t, u^*)$, or $t^*[\vec{X} \leftarrow u^*]$ at depth n, the subterms t and u^* are at depth $n+1$. The expression t^* stays at depth n because of the congruence rule. If $t^* = w^*[\phi][\psi] \sim w^*[\psi][\phi]$, we want the path to these closures to be of the same length. For distributors, intuitively we see $\mathcal{A}y.u^p$ as being at depth $n+1$, so we consider u^p to be at depth $n+2$. Only the depths of the closures appear in the output.

- $\overline{0}\chi$
- $\overline{0}\langle\ \rangle$
- $\mathcal{A}x.(\underline{n+1}t)$
- $@(\underline{n+1}t, \underline{n+1}u)$
- $(\underline{n}t^*)(\overline{n+1}[x_1, \ldots, x_p \leftarrow \underline{n+1}u])$
- $(\underline{n}t^*)(\overline{n+1}[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.\underline{n+2}u^p])$
- $\underline{\langle n+1,\ldots,n+1 \rangle}t^p$

For an expression t^*, $\partial(t^*, n) = \{k^{m(k)} \mid k \in \mathbb{N}\}$ means that when t^* is of depth n, there are $m(k)$ closures of t^* of depth k. The signature of the depth function is as follows:

$$\partial : \underbrace{\mathcal{T}^*}_{\text{term, stream, tuple } t^*} \rightarrow \underbrace{\mathbb{N}^p}_{\text{input depths}} \rightarrow \underbrace{\mathcal{M}_f(\mathbb{N})}_{\text{output depths}}$$

Let t^* be a term, stream or a tuple, let τ be a term or a stream, let χ, χ_i be variables. For any expression t^*, we measure $\partial(t^*, 1)$.

We now define the *depth* $\partial(t^*, n)$, which measures a term t^* that is at depth n. We define it by induction on t^*:

Definition 5.4.1. [Depth] The other cases are similar to the definition for the weakening calculus.

- $\partial(\langle\, t_1, \ldots, t_p \,\rangle, n) = (\sum_{i=1}^{p} m_i)$
  where $\partial(t_i, n+1) = m_i$, and $p \geq 0$.
- $\partial(t^*[x_1, \ldots, x_p \twoheadleftarrow \mathcal{A}y.u^p], n) = \partial(t^*, n) + \partial(u^p, n+2) + \{n\}$ $(p \geq 0)$.

From the following lemma we deduce that depth is monotonous, i.e. if $n \leq m, \partial(t^*, n) \leq \partial(t^*, m)$.

Lemma 5.4.2. For $t^* \in \Lambda\mu S_a, n \in \mathbb{N}$, $\partial(t^*, n) \leq \partial(t^*, n+1)$.

Proof. By induction on t^*.

- $\partial(\chi, n) = \emptyset = \partial(\chi, n+1)$.
- $$\begin{aligned} \partial(\langle\, t_1, \ldots, t_p \,\rangle, n) &= (\sum_{i=1}^{p} m_i) \\ &\leq (\sum_{i=1}^{p} m'_i) \\ &= \partial(\langle\, t_1, \ldots, t_p \,\rangle, n+1) \end{aligned}$$

  where $\partial(t_i, n+1) = m_i, \partial(t_i, n+2) = m'_i$, and $p \geq 0$.

- $$\begin{aligned}\partial(@(t,\tau),n) &= \partial(t,n+1)+\partial(u,n+1)\\ &\leq \partial(t,n+2)+\partial(u,n+2)\\ &= \partial(@(t,\tau),n+1)\end{aligned}$$

- $$\begin{aligned}\partial(\mathcal{A}x.t,n) &= \partial(t,n+1)\\ \leq \partial(t,n+2) &= \partial(\mathcal{A}x.t,n+1)\end{aligned}$$

- $$\begin{aligned}\partial(t^*[\chi_1,\ldots,\chi_p \leftarrow \tau],n) &= \partial(t^*,n)+\partial(\tau,n+1)+\{n\}\\ \leq \partial(t^*,n+1)+\partial(\tau,n+2)+\{n+1\} &= \partial(t^*[\chi_1,\ldots,\chi_p \leftarrow \tau],n+1)\end{aligned}$$

  where $p \geq 0$.

- $$\begin{aligned}\partial(t^*[x_1,\ldots,x_p \twoheadleftarrow \mathcal{A}y.u^p],n) &= \partial(t^*,n)+\partial(u^p,n+2)+\{n\}\\ \leq \partial(t^*,n+1)+\partial(u^p,n+3)+\{n+1\} &= \partial(t^*[x_1,\ldots,x_p \twoheadleftarrow \mathcal{A}y.u^p],n+1)\end{aligned}$$

  where $p \geq 0$.

□

5.5 Strong normalization of $\longrightarrow_s$

We now show that $\longrightarrow_s$ is strongly normalizing, using our measure. For $t \in \Lambda\mu S_a$, the measure is such that, for a step $t \longrightarrow_s u$:

1. There is a weakening step (M1):

$$[\![\, t \,]\!]_w \longrightarrow_w [\![\, u \,]\!]_w$$

Otherwise,

$$[\![t]\!]_w = [\![u]\!]_w$$

2. The weight decreases (M2):

$$m_t > m_u$$

where

$$\omega(t, 1) = (m_t, f_t)$$
$$\omega(u, 1) = (m_u, f_u)$$

Otherwise,

$$m_t = m_u$$

3. The number of closures decreases, otherwise it remains constant. (M3)
4. The depth decreases (M4):

$$\partial(t, 1) > \partial(u, 1)$$

The following list summarizes all the different cases.

- Lifting rules:
  - Lifting a weakened sharing out of a weakened distributor: M4
  - Lifting a weakened sharing out of anything else: M1
  - Lifting a non-weakened sharing out of anything: M4
  - Lifting a distributor out of anything: M4
- Compounding rules:
  - Compounding when the first sharing is a weakening: M1
  - Compounding when both sharings are not weakenings: M3
- Subtitution rule: M3
- Duplication rules:

– Duplicating a weakened application/abstraction: M1

– Removing a weakened distributor: M1

– Duplicating a non-weakened application/abstraction: M2

– Removing a non-weakened distributor: M2

5.5.1 Measure 1: weakening reduction

Lemma 5.5.1. The following reduction steps translate to a reduction step in the weakening calculus (M1):

- Lifting weakened sharings. Let $[\phi]$ be a weakened sharing:
  - $\mathcal{A}x.(u[\phi]) \longrightarrow_s (\mathcal{A}x.u)[\phi]$ if $x \in FV(u)$
  - $\begin{array}{l} @(u[\phi], t) \\ @(u, t[\phi]) \end{array} \longrightarrow_s @(u, t)[\phi]$
  - $u^*[\vec{\chi_p} \leftarrow \tau[\phi]] \longrightarrow_s u^*[\vec{\chi_p} \leftarrow \tau][\phi]$
  - For $p \neq 0$:
    $u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi][\phi]] \longrightarrow_s u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi]][\phi]$ if $y \in FV(\langle\, \vec{t_p} \,\rangle[\Psi])$
- Duplications on weakenings:
  - $u^*[\leftarrow @(v, \tau)] \longrightarrow_s u^*[\leftarrow v][\leftarrow \tau]$
  - $u^*[\leftarrow \mathcal{A}x.t] \longrightarrow_s u^*[\twoheadleftarrow \mathcal{A}x.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t]]$
  - $u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\leftarrow y]] \longrightarrow_s u^*$
- Compounding if the first sharing is a weakening:

$$u^*[\leftarrow \chi][\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau] \longrightarrow_s u^*[\vec{\chi'_m}, \vec{\chi''_n} \leftarrow \tau]$$

Proof. We show that each case corresponds to a step in the weakening calculus.

- Lifting a weakened sharing from abstractions (similar for applications, sharings):

$$\mathcal{A}x.([\![\, u \,]\!]_w\{\!|\, \phi \,|\!\}_w) \longrightarrow_w (\mathcal{A}x.[\![\, u \,]\!]_w)\{\!|\, \phi \,|\!\}_w$$

- Lifting a weakened sharing from a non-weakened distributor:

$$[\![\, u^* \,]\!]_w\{\mathcal{A}y.[\![\, t_i[\Psi][\phi] \,]\!]_w/x_i\}_{1\leq i\leq p} \longrightarrow_w^+ [\![\, u^* \,]\!]_w\{\mathcal{A}y.[\![\, t_i[\Psi] \,]\!]_w/x_i\}_{1\leq i\leq p}\{\!|\, \phi \,|\!\}_w$$

- For duplication of applications:

$$\begin{aligned}[\![\, u^*[\leftarrow @(v,\tau)] \,]\!]_w &= [\![\, u^* \,]\!]_w[\leftarrow @([\![\, v \,]\!]_w, [\![\, \tau \,]\!]_w)] \\ &\longrightarrow_w [\![\, u^* \,]\!]_w[\leftarrow [\![\, v \,]\!]_w][\leftarrow [\![\, \tau \,]\!]_w] \\ &= [\![\, u^*[\leftarrow v][\leftarrow \tau] \,]\!]_w\end{aligned}$$

- For duplication of abstractions:

$$\begin{aligned}[\![\, u^* \,]\!]_w[\leftarrow \mathcal{A}x.[\![\, t \,]\!]_w] &\longrightarrow_w [\![\, u^* \,]\!]_w[\leftarrow [\![\, t \,]\!]_w\{\bullet/x\}] \\ &= [\![\, u^* \,]\!]_w\{\!|\, \twoheadleftarrow \mathcal{A}x.\langle\ \rangle[\leftarrow t] \,|\!\}_w\end{aligned}$$

- For distributor removal:

$$\begin{aligned}[\![\, u^* \,]\!]_w\{\!|\, \twoheadleftarrow \lambda y.\langle\ \rangle[\leftarrow y] \,|\!\}_w &= [\![\, u^* \,]\!]_w[\leftarrow \bullet] \\ &\longrightarrow_w [\![\, u^* \,]\!]_w\end{aligned}$$

- Compounding when the first sharing is a weakening:

$$u^*[\leftarrow \chi][\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau] \longrightarrow_s u^*[\vec{\chi'_m}, \vec{\chi''_n} \leftarrow \tau]$$

$$[\![\, u^* \,]\!]_w\{\!|\, \tau/\vec{\chi'_m}, \vec{\chi''_n} \,|\!\}_w[\leftarrow [\![\, \tau \,]\!]_w] \longrightarrow_w [\![\, u^* \,]\!]_w\{\!|\, \tau/\vec{\chi'_m}, \vec{\chi''_n} \,|\!\}_w$$

since $[\![\, \tau \,]\!]_w$ is a subterm of $[\![\, u^* \,]\!]_w\{\!|\, \tau/\vec{\chi'_m}, \vec{\chi''_n} \,|\!\}_w$.

□

5.5.2 Measure 2: weight

Lemma 5.5.2. The following reduction steps are such that, if $t^* \longrightarrow_s u^*$:

1. Their denotation in the weakening calculus is unchanged, i.e. $[\![\, t^* \,]\!]_w = [\![\, u^* \,]\!]_w$
2. Their weight strictly decreases, i.e. $m_{t^*} > m_{u^*}$ if $\omega(t^*, \vec{n_p}) = (m_{t^*}, f_{t^*})$ and $\omega(u^*, \vec{n_p}) = (m_{u^*}, f_{u^*})$.

- Duplications on non-weakenings
  - $u^*[\vec{\chi_p} \leftarrow @(v,\tau)] \longrightarrow_s u^*\{@(y_i, z_i)/x_i\}[\vec{y_p} \leftarrow v][\vec{z_p} \leftarrow \tau]$

– $u^*[\vec{x_p} \leftarrow \mathcal{A}x.t] \longrightarrow_s u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}x.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow t]]$

– $u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\vec{z_q} \leftarrow y]] \longrightarrow_s u^*\{(\mathcal{A}y_i.t_i[\vec{z_i}^{l_i} \leftarrow y_i])/x_i\}$
and $\{z_i^1, \ldots, z_i^{l_i}\} = \{\vec{z_q}\} \cap FV(t_i)$

Proof. For M1, the weakenings are the same for terms on the LHS and RHS of the reductions, so there are no weakening reductions. We show that the weight, measure 2, strictly decreases.

- When duplicating applications (similar for abstractions), the weight(M2) strictly decreases:

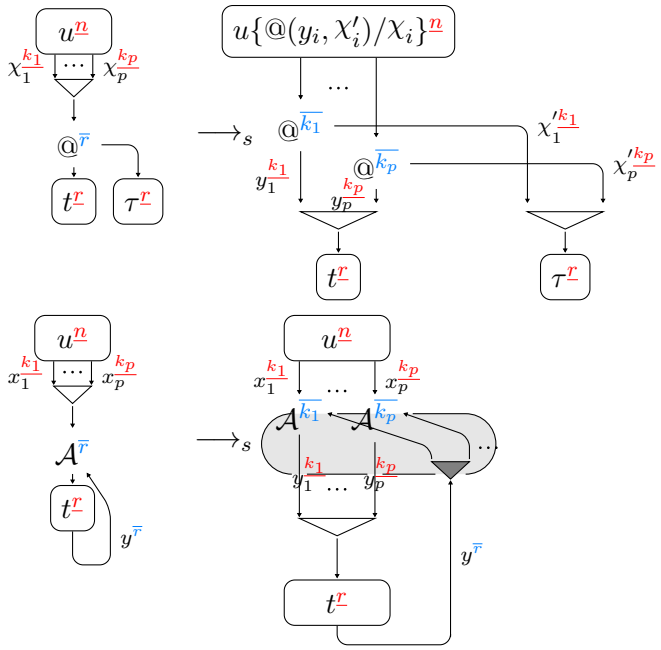
$$
\begin{aligned}
&\omega(u^*[\vec{\chi_p} \leftarrow @(v,\tau)], n_1, \ldots, n_q)\\
&= (m + m_v + m_\tau + \{r\}, (f\backslash\chi_1, \ldots, \chi_p) + f_v + f_\tau)\\
&\omega(u^*\{@(y_1, \chi_1')/\chi_1\} \ldots \{@(y_p, \chi_p')/\chi_p\}[\vec{y_p} \leftarrow v][\vec{\chi_p'} \leftarrow \tau], n_1, \ldots, n_q)\\
&= (m + \sum_{i=1}^{p}(m_y^i + m_{\chi'}^i) + \{f(\chi_i) \mid i \leq p\} + m_v + m_\tau,\\
&((f\backslash\vec{\chi_p}) + \sum_{i=1}^{p}(f_y^i + f_{\chi'}^i))\backslash\vec{y_p}, \vec{\chi_p'} + f_v + f_\tau)\\
&= (m + \{f(\chi_i) \mid i \leq p\} + m_v + m_\tau, (f\backslash\vec{\chi_p}) + f_v + f_\tau)
\end{aligned}
$$

where

$$
\begin{aligned}
\omega(u^*, n_1, \ldots, n_q) &= (m, f)\\
\omega(y_i, f(\chi_i)) &= (m_y^i, f_y^i) = (\emptyset, \{y_i^{f(\chi_i)}\})\\
\omega(\chi_i', f(\chi_i)) &= (m_{\chi'}^i, f_{\chi'}^i) = (\emptyset, \{\chi_i'^{f(\chi_i)}\}\\
\omega(\tau, r) &= (m_\tau, f_\tau)\\
\omega(v, r) &= (m_v, f_v)
\end{aligned}
$$

where $r = \sum_{i=1}^{q} f(\chi_i)$.

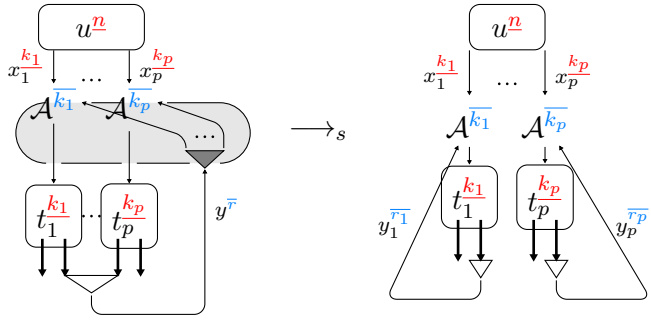
using Lemma 5.3.4 on substitutions. The weight strictly decreases since $r > f(\chi_i)$. Graphically (with $k_i = f(\chi_i)$) we have:



- When removing the distributor, the weight(M2) strictly decreases:

$$\omega(u^*, n) = (m, f)$$
$$\omega(t_i, f(x_i)) = (m^i, f^i)$$
$$\omega(\langle\, \vec{t_p} \,\rangle, f(x_1), \ldots, f(x_p)) = (m', f') = (\sum_{i=1}^{p} m^i, \sum_{i=1}^{p} f^i)$$
$$\omega(y, r) = (\emptyset, \{y^r\}) \text{ where } r = \sum_{j=1}^{q} f'(z_j)$$
$$\omega(\langle\, \vec{t_p} \,\rangle[\vec{z_q} \leftarrow y]], f(x_1), \ldots, f(x_p)) = (m' + \{r\}, f' \backslash \vec{z_q})$$
$$\omega(y_i, r_i) = (\emptyset, \{y_i^{r_i}\}) \text{ where } r_i = \sum_{k=1}^{l_i} f^i(z_k)$$
$$\omega((\mathcal{A}y_i.t_i[\vec{z_i}^{l_i} \leftarrow y_i]), f(x_i)) = (m^i + \{f(x_i)\} + \{r_i \mid 1 \le i \le p\}, f^i \backslash \vec{z}^{l_i})$$
$$\omega(u^*[\vec{x_p} \nleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\vec{z_q} \leftarrow y]], n) = (m + m' + \{f(x_i) \mid 1 \le i \le p\} + \{r\},$$
$$(f \backslash \vec{x_p}) + (f' \backslash \vec{z_q}))$$
$$\omega(u^*\{(\mathcal{A}y_i.t_i[\vec{z_i}^{l_i} \leftarrow y_i])/x_i\}, n) = (m + m' + \{f(x_i) \mid 1 \le i \le p\} +$$
$$\{r_i \mid 1 \le i \le p\},$$
$$(f \backslash \vec{x_p}) + \sum_{i=1}^{i} (f^i \backslash \vec{z_i}^{l_i}))$$

using Lemma 5.3.4 on substitutions. Since $r_i < r$, the weight strictly decreases. Graphically (with $k_i = f(x_i)$) we have:



□

5.5.3 Measure 3: number of closures

The following reduction steps are such that, if $t^* \longrightarrow_s u^*$:

1. Their denotation in the weakening calculus is unchanged, i.e. $[\![t^*]\!]_w = [\![u^*]\!]_w$
2. Their weight is unchanged, i.e. $\omega(t^*, \vec{n_p}) = \omega(u^*, \vec{n_p})$
3. The number of closures strictly decreases.

The rules that strictly decrease the number of closures are:

- Compounding rule when both sharings are not weakenings

$$u^*[\vec{\chi_p} \leftarrow \chi][\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau] \longrightarrow_s u^*[\vec{\chi'_m}, \vec{\chi_p}, \vec{\chi''_n} \leftarrow \tau]$$

- Substitution rule $u^*[x \leftarrow \tau] \longrightarrow_s u^*\{\tau/x\}$

Proof. For M1, the weakenings are the same for terms on the LHS and RHS of the reductions, so there are no weakening reductions.

- For the compounding rule, the weight (M2) is not affected:

– $u^*[\vec{\chi_p} \leftarrow \chi][\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau] \longrightarrow_s u^*[\vec{\chi'_m}, \vec{\chi_p}, \vec{\chi''_n} \leftarrow \tau]$

$$\omega(\chi, s) = (\emptyset, \{\chi^s\})$$
$$\text{where } s = \sum_{l=1}^{p} f(\chi_l)$$
$$\omega(u^*, n) = (m, f)$$
$$\omega(u^*[\vec{\chi_p} \leftarrow \chi], n) = (m, f \backslash \vec{\chi_p} + \{\chi^s\})$$
$$\omega(\tau, r) = (m^\tau, f^\tau) \text{ where } r = \sum_{j=1}^{m} f(\chi'_j) + \sum_{k=1}^{n} f(\chi''_k) + s$$
$$\omega((u^*[\vec{\chi_p} \leftarrow \chi])[\vec{\chi'_m}, \chi, \vec{\chi''_n} \leftarrow \tau], n)$$
$$= (m + m^\tau, (f \backslash \vec{\chi_p}, \vec{\chi'_m}, \vec{\chi''_n}) + f^\tau)$$
$$\omega(u^*[\vec{\chi'_m}, \vec{\chi_p}, \vec{\chi''_n} \leftarrow \tau], n)$$
$$= (m + m^\tau, (f \backslash \vec{\chi_p}, \vec{\chi'_m}), \vec{\chi''_n} + f^\tau)$$

The compounding rule reduces the number of sharings (M3).

- For the substitution rule, the weight (M2) is not affected:

$$\begin{aligned} \omega(u^*, n) &= (m, f) \\ \omega(\tau, r) &= (m^\tau, f^\tau) \text{ where } r = f(x) \\ \omega(u^*[x \leftarrow \tau], n) &= (m + m^\tau, (f \backslash x) + f^\tau) \\ &= \omega(u^*\{\tau/x\}, n) \end{aligned}$$

Using Lemma 5.3.4 on substitutions and Corollary 5.3.5. However, the substitution rule strictly reduces the number of sharings (M3).

□

5.5.4 Measure 4: depth

The following reduction steps are such that, if $t^* \longrightarrow_s u^*$:

1. Their denotation in the weakening calculus is unchanged, i.e. $[\![t^*]\!]_w = [\![u^*]\!]_w$
2. Their weight is unchanged, i.e. $\omega(t^*, \vec{n_p}) = \omega(u^*, \vec{n_p})$
3. The number of closures remains the same.

4. Their depth strictly decreases, i.e. $\partial(t^*, n) > \partial(u^*, n)$.

Depth decreases in the following situations:

- Lifting distributors and non-weakened sharings out of anything, except non-weakened distributors.

  Let $[\phi]$ be a distributor or a non-weakened sharing:

  - $\mathcal{A}x.(u[\phi]) \longrightarrow_s (\mathcal{A}x.u)[\phi]$ if $x \in FV(u)$
  - $\begin{matrix} @(u[\phi], t) \\ @(u, t[\phi]) \end{matrix} \longrightarrow_s @(u, t)[\phi]$
  - $u^*[\vec{\chi_p} \leftarrow \tau[\phi]] \longrightarrow_s u^*[\vec{\chi_p} \leftarrow \tau][\phi]$
  - $u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\phi]] \longrightarrow_s u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\phi]$ if $y \in FV([\Psi])$

- Lifting a distributor out of a non-weakened distributor.Let $[\phi]$ be a distributor: For $p > 0$,

  $$u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p}\,\rangle[\Psi][\phi]] \longrightarrow_s u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p}\,\rangle[\Psi]][\phi]$$

  if $y \in FV(\langle\, \vec{t_p}\,\rangle[\Psi])$

- Lifting a weakened sharing out of a weakened distributor:

  $$u^*[\twoheadleftarrow \mathcal{A}y.t^0[\leftarrow \tau]] \longrightarrow_s u^*[\twoheadleftarrow \mathcal{A}y.t^0][\leftarrow \tau]$$

- Lifting a non-weakened sharing out of a non-weakened distributor. Let $p, q > 0$:

  $$u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p[\chi_q \leftarrow \tau]] \longrightarrow_s u^*[\vec{x_p} \twoheadleftarrow \mathcal{A}y.t^p][\chi_q \leftarrow \tau]$$

Proof. • Lifting a weakened distributor out of an abstraction (other cases are similar):

  For M1, the weakenings are the same for terms on the LHS and RHS of the reductions, so there are no weakening reductions.

  $$\mathcal{A}x.[\![\, u \,]\!]_w\{\bullet/y\}\{\!|\, \Psi \,|\!\}_w = (\mathcal{A}x.[\![\, u \,]\!]_w)\{\bullet/y\}\{\!|\, \Psi \,|\!\}_w$$

The weight (M2) is not affected:

$$\begin{aligned}
\omega(\langle\ \rangle[\Psi], n) &= (m^{\langle\ \rangle[\Psi]}, f^{\langle\ \rangle[\Psi]}) \\
\omega(u, n) &= (m^u, f^u) \\
\omega(u[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]], n) &= (m^u + m^{\langle\ \rangle[\Psi]} + \{n\} + \{f^{\langle\ \rangle[\Psi]}(y)\}, \\
&\quad f^u + (f^{\langle\ \rangle[\Psi]} \backslash y)) \\
\omega(\mathcal{A}x.u, n) &= (m^u + \{n\} + \{f^u(x)\}, f^u \backslash x) \\
\omega(\langle\ \rangle[\Psi], n) &= (m^{\langle\ \rangle[\Psi]}, f^{\langle\ \rangle[\Psi]}) \\
\omega(\mathcal{A}x.(u[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]), n) &= (m^u + m^{\langle\ \rangle[\Psi]} + \{n\} + \{f^{\langle\ \rangle[\Psi]}(y)\} \\
&\quad + \{n\} + \{f^u(x)\}, f^u \backslash x + f^{\langle\ \rangle[\Psi]} \backslash y) \\
\omega((\mathcal{A}x.u)[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]], n) &= (m^u + \{n\} + \{f^u(x)\} \\
&\quad + m^{\langle\ \rangle[\Psi]} + \{n\} + \{f^{\langle\ \rangle[\Psi]}(y)\}, \\
&\quad f^u \backslash x + f^{\langle\ \rangle[\Psi]} \backslash y)
\end{aligned}$$

The number of closures (M3) stays the same.

Depth (M4) strictly decreases:

$$\begin{aligned}
\partial(\mathcal{A}x.(u[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]), n) &= \partial(u[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]]), n+1) \\
&= \partial(u, n+1) + \partial(\langle\ \rangle[\Psi], n+3) + \{n+1\}
\end{aligned}$$

$$\begin{aligned}
\partial((\mathcal{A}x.u)[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]], n) &= \partial(\mathcal{A}x.u, n) + \partial(\langle\ \rangle[\Psi], n+2) + \{n\} \\
&= \partial(u, n+1) + \partial(\langle\ \rangle[\Psi], n+2) + \{n\}
\end{aligned}$$

Using the monotonicity Lemma 5.4.2 $\partial(\langle \vec{t_p} \rangle, n+2) \leq \partial(\langle \vec{t_p} \rangle, n+3)$, and the depth strictly decreases.

- Lifting a non-weakened distributor out of an abstraction (other cases are similar):

$$\mathcal{A}x.(u[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle \vec{t_p} \rangle[\Psi]]) \longrightarrow_s (\mathcal{A}x.u)[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle \vec{t_p} \rangle[\Psi]]$$

For M1, the weakenings are the same for terms on the LHS and RHS of the reductions, so there are no weakening reductions.

The weight (M2) is not affected:

$$\begin{aligned}
\omega(u, n) &= (m^u, f^u) \\
\omega(\langle\, \vec{t_p} \,\rangle[\Psi], f(x_1), \ldots, f(x_p)) &= (m', f') \\
\omega((u[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi]]), n) &= (m^u + m' + \{f(x_i) \mid 1 \leq i \leq p\} + \{f'(y)\}, \\
&\quad (f^u \backslash \vec{x_p}) + (f' \backslash y)) \\
\omega(\mathcal{A}x.u, n) &= (\{n\} + m^u + \{f^u(x)\}, f^u \backslash x) \\
\omega(\mathcal{A}x.(u[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi]]), n) &= (\{n\} + m^u + m' + \{f(x_i) \mid 1 \leq i \leq p\} + \\
&\quad \{f'(y)\} + \{f^u(x)\}, \\
&\quad (f^u \backslash \vec{x_p}, x) + (f' \backslash y)) \\
\omega((\mathcal{A}x.u)[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi]], n) &= (\{n\} + m^u + \{f^u(x)\} + \{f(x_i) \mid 1 \leq i \leq p\} + \\
&\quad \{f'(y)\} + m', \\
&\quad f^u \backslash x, \vec{x_p} + (f' \backslash y))
\end{aligned}$$

The number of closures stays the same.

Depth (M4) strictly decreases:

$$\begin{aligned}
\partial(\mathcal{A}x.(u[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi]]), n) &= \partial((u[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi]]), n+1) \\
&= \partial(u, n+1) + \partial(\langle\, \vec{t_p} \,\rangle[\Psi], n+3) + \{n+1\}
\end{aligned}$$

$$\begin{aligned}
\partial((\mathcal{A}x.u)[\vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle\, \vec{t_p} \,\rangle[\Psi]], n) &= \partial(\mathcal{A}x.u, n) + \partial(\langle\, \vec{t_p} \,\rangle[\Psi], n+2) + \{n\} \\
&= \partial(u, n+1) + \partial(\langle\, \vec{t_p} \,\rangle[\Psi], n+2) + \{n\}
\end{aligned}$$

Using Lemma 5.4.2 (monotonicity) $\partial(\langle\, \vec{t_p} \,\rangle, n+2) \leq \partial(\langle\, \vec{t_p} \,\rangle, n+3)$, and the depth strictly decreases.

- Lifting a non-weakened sharing out of an abstraction (other cases are similar)

  For M1, the weakenings are the same for terms on the LHS and RHS of the reductions, so there are no weakening reductions.

The weight (M2) is not affected:

$$\begin{aligned}
\omega(\tau, r) &= (m^\tau, f^\tau) \\
\omega(u^*, n) &= (m, f) \\
\omega((u[\vec{\chi_p} \leftarrow \tau]), n) &= (m + m^\tau, f \backslash \vec{\chi_p} + f^\tau) \\
\omega(\mathcal{A}x.u, n) &= (m, f \backslash x) \\
\omega(\mathcal{A}x.(u[\vec{\chi_p} \leftarrow \tau]), n) &= (m + m^\tau, f \backslash \vec{\chi_p}, x + f^\tau) \\
\omega((\mathcal{A}x.u)[\vec{\chi_p} \leftarrow \tau], n) &= (m + m^\tau, f \backslash \vec{\chi_p}, x + f^\tau)
\end{aligned}$$

The number of closures does not change.

Depth (D4) strictly decreases:

$$\begin{aligned}
\partial(\mathcal{A}x.(u[\vec{\chi_p} \leftarrow \tau]), n) &= \partial((u[\vec{\chi_p} \leftarrow \tau]), n + 1) \\
&= \partial(u, n + 1) + \partial(\tau, n + 2) + \{n + 1\}
\end{aligned}$$

$$\begin{aligned}
\partial((\mathcal{A}x.u)[\vec{\chi_p} \leftarrow \tau], n) &= \partial((\mathcal{A}x.u), n) + \partial(\tau, n + 1) + \{n\} \\
&= \partial(u, n + 1) + \partial(\tau, n + 1) + \{n\}
\end{aligned}$$

Using Lemma 5.4.2 (monotonicity) $\partial(\tau, n + 1) \leq \partial(\tau, n + 2)$, and the depth strictly decreases.

- Lifting weakened sharings from weakened distributors:

  For M1, the weakenings are the same for terms on the LHS and RHS of the reductions, so there are no weakening reductions.

$$\begin{aligned}
u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\leftarrow \tau]] &\longrightarrow_s u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\leftarrow \tau] \\
[\![\, u^* \,]\!]_w \{\!|\, \psi \,|\!\}_w \{\bullet / y\}[\leftarrow \tau] &= [\![\, u^* \,]\!]_w \{\!|\, \psi \,|\!\}_w \{\bullet / y\}[\leftarrow \tau]
\end{aligned}$$

The weight (M2) is not affected:

$$\begin{aligned}
\omega(\langle\ \rangle[\Psi], n) &= (m^{\langle\ \rangle[\Psi]}, f^{\langle\ \rangle[\Psi]}) \\
\omega(u^*, n) &= (m, f) \\
\omega(\tau, n) &= (m^\tau, f^\tau) \\
\omega(u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\leftarrow \tau]], n) &= (m + m^{\langle\ \rangle[\Psi]} + \{n\} + \{f^{\langle\ \rangle[\Psi]}(y)\} + m^\tau, \\
&\qquad f + (f^{\langle\ \rangle[\Psi]} \backslash y) + f^\tau) \\
\omega(u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\leftarrow \tau], n) &= (m + m^{\langle\ \rangle[\Psi]} + \{n\} + \{f^{\langle\ \rangle[\Psi]}(y)\} + m^\tau, \\
&\qquad f + (f^{\langle\ \rangle[\Psi]} \backslash y) + f^\tau)
\end{aligned}$$

The number of closures (M3) remains the same.

The depth strictly decreases:

$$\begin{aligned}
&\partial(u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi][\leftarrow \tau]], n) = \\
&= \partial(u^*, n) + \partial(\langle\ \rangle[\Psi][\leftarrow \tau]], n+2) + \{n+1\} \\
&= \partial(u^*, n) + \partial(\langle\ \rangle[\Psi], n+2) + \partial(\tau, n+3) + \{n+1\} + \{n+3\}
\end{aligned}$$

$$\begin{aligned}
&\partial(u^*[\twoheadleftarrow \mathcal{A}y.\langle\ \rangle[\Psi]][\leftarrow \tau], n) \\
&= \partial(u^*, n) + \partial(\langle\ \rangle[\Psi]], n+2) + \partial(\tau, n+1) + \{n+1\} + \{n+1\}
\end{aligned}$$

By monotonicity of depth.

□

Our measure strictly decreases after each sharing reduction, we therefore conclude:

Theorem 5.5.3. The reduction $\longrightarrow_s$ is strongly normalizing.

Chapter 6

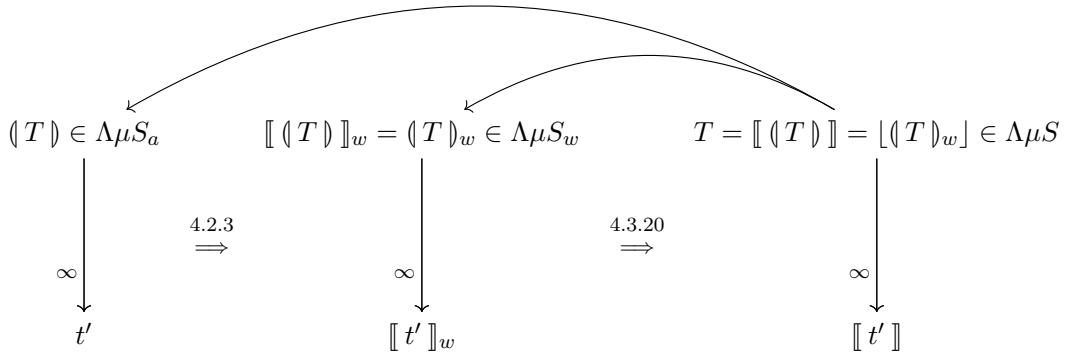
Proof of PSN and confluence

Using the results from the previous chapters, we show PSN for the atomic $\lambda\mu$-calculus with respect to the $\Lambda\mu S$-calculus. A corollary that follows is that the atomic $\lambda\mu$-calculus is confluent.

6.1 Preservation of strong normalization

We now combine Theorems 4.2.3 ($[\![-]\!]_w$ preserves non-termination), 4.3.20 (PSN for the weakening calculus), and strong normalization of sharing reductions 5.5.3 to get PSN:

Theorem 6.1.1 (PSN for the atomic $\lambda\mu$-calculus). For $T \in \Lambda\mu S$, if $(\!|\, T \,|\!) \in \Lambda\mu S_a$ has an infinite reduction path, then $T \in \Lambda\mu S$ has an infinite reduction path.

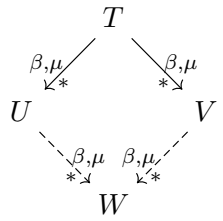


Proof. Let $T \in \Lambda\mu S$. Suppose there exists an infinite reduction path of $(\!|\, T \,|\!)$. Because of Theorem 5.5.3, infinite reductions must come from (β, μ)-reductions. From Theorem 4.2.3, from $(\!|\, T \,|\!)$ we can construct an infinite (β, μ)-reduction path of $[\![\, (\!|\, T \,|\!) \,]\!]_w$. By Lemma 4.1.5, $[\![\, (\!|\, T \,|\!) \,]\!]_w = (\!|\, T \,|\!)_w$. From Theorem 4.3.20, from $(\!|\, T \,|\!)_w$ we can construct an infinite (β, μ)-reduction path of T. □

6.2 Corollary: confluence

Using lemmas showing PSN, we can deduce confluence for the atomic $\lambda\mu$-calculus.

Theorem 6.2.1 (Confluence of $\Lambda\mu S$). Let $T, U, V \in \Lambda\mu S$. If $T \longrightarrow^*_{\beta,\mu} U$ and $T \longrightarrow^*_{\beta,\mu} V$, there exists $W \in \Lambda\mu S$ such that $U, V \longrightarrow^*_{\beta,\mu} W$.



Proof. (sketch.) Let $\longrightarrow_{\beta,\mu}$ denote one of the reduction rules in $\Lambda\mu S$. Let M, M', N, N' be terms of $\Lambda\mu S$, and $\mathcal{S}, \mathcal{S}' \in \Lambda\mu S$ be streams. The proof is similar to that of the $\lambda\mu$-calculus, using a new reduction $\twoheadrightarrow_{\beta,\mu}$ (the *parallel one-step reduction*), such that $\longrightarrow^*_{\beta,\mu}$ is the transitive closure of $\twoheadrightarrow_{\beta,\mu}$, and that $\twoheadrightarrow_{\beta,\mu}$ is confluent. The reduction $\twoheadrightarrow_{\beta,\mu}$ is defined as follows:

- $\chi \twoheadrightarrow_{\beta,\mu} \chi$
- If $M \twoheadrightarrow_{\beta,\mu} M'$, then $\mathcal{A}x.M \twoheadrightarrow_{\beta,\mu} \mathcal{A}x.M'$
- If $M \twoheadrightarrow_{\beta,\mu} M'$ and $N^* \twoheadrightarrow_{\beta,\mu} N'^*$, then $@(M, N^*) \twoheadrightarrow_{\beta,\mu} @(M', N'^*)$
- If $M \twoheadrightarrow_{\beta,\mu} M'$ and $N \twoheadrightarrow_{\beta,\mu} N'$, then $(\lambda x.M)N \twoheadrightarrow_{\beta,\mu} M'\{N'/x\}$
- If $M \twoheadrightarrow_{\beta,\mu} M'$ and $N \circ \mathcal{S} \twoheadrightarrow_{\beta,\mu} N' \circ \mathcal{S}'$, then $(\lambda x.M)(N \circ \mathcal{S}) \twoheadrightarrow_{\beta,\mu} M'\{N'/x\}\mathcal{S}'$
- If $M \twoheadrightarrow_{\beta,\mu} M'$ and $N \twoheadrightarrow_{\beta,\mu} N'$, then $(\mu\beta.M)N \twoheadrightarrow_{\beta,\mu} \mu\beta.M'\{N' \circ \beta/\beta\}$
- If $M \twoheadrightarrow_{\beta,\mu} M'$ and $\mathcal{S} \twoheadrightarrow_{\beta,\mu} \mathcal{S}'$, then $(\mu\beta.M)\mathcal{S} \twoheadrightarrow_{\beta,\mu} M'\{\mathcal{S}'/\beta\}$

We then define the maximal parallel one-step reduct:

$$\begin{aligned}
\chi\Downarrow &= \chi \\
(\mathcal{A}x.M)\Downarrow &= \mathcal{A}x.M\Downarrow \\
((\lambda x.M)N)\Downarrow &= M\Downarrow\{N\Downarrow/x\} \\
((\lambda x.M)(N \circ \mathcal{S}))\Downarrow &= (M\Downarrow\{N\Downarrow/x\})\mathcal{S}\Downarrow \\
((\mu\beta.M)N)\Downarrow &= \mu\beta.M\Downarrow\{(N\Downarrow \circ \beta)/\beta\} \\
((\mu\beta.M)\mathcal{S})\Downarrow &= M\Downarrow\{\mathcal{S}\Downarrow/\beta\} \\
@(M,N) &= @(M\Downarrow, N\Downarrow) \text{ if we do not have a redex}
\end{aligned}$$

□

The proof for confluence for $\Lambda\mu S_a$ uses the two lemmas below: atomic translations of $\Lambda\mu S$-terms are sharing normal, and atomic reduction simulates $\Lambda\mu S$-reduction.

Lemma 6.2.2 (Atomic translations of $\Lambda\mu S$-terms are $\longrightarrow_s$-normal). If $t \in \Lambda\mu S_a$ is sharing normal then $(\!|\, [\![\, t \,]\!] \,|\!) = t$.

Proof. By induction. Let U be a term or a stream, let χ, χ_i be variables. Recall that $\frac{l_i}{\chi_i}$ means replacing χ_i with l_i fresh distinct variables. Let $\sigma = \underbrace{\frac{l_1}{\chi_1} \ldots \frac{l_p}{\chi_p}}_{\text{occurrences in } U}$, replacing the different occurrences of each free variable χ_i with fresh, distinct variables. Let $\Phi = [\vec{\chi_1}^{\,l_1} \leftarrow \chi_1] \ldots [\vec{\chi_p}^{\,l_p} \leftarrow \chi_p]$. We use the inductive hypothesis: $(\!|\, [\![\, u^* \,]\!]\sigma \,|\!)' = u^*\sigma$ if u^* is sharing normal.

- $[\![\, \chi \,]\!] = \chi$. Then $(\!|\, [\![\, \chi \,]\!] \,|\!) = \chi$.
- $[\![\, \mathcal{A}x.t \,]\!] = \mathcal{A}x.[\![\, t \,]\!]$ Then:

$$\begin{aligned}
(\!|\, [\![\, \mathcal{A}x.t \,]\!]\sigma \,|\!)' &= (\!|\, \mathcal{A}x.[\![\, t \,]\!]\sigma \,|\!)' \\
&= \mathcal{A}x.(\!|\, [\![\, t \,]\!]\sigma \,|\!)' \\
&= \mathcal{A}x.t\sigma \\
&= (\mathcal{A}x.t)\sigma
\end{aligned}$$

  by induction on t.

- $[\![@(t,\tau)]\!] = @([\![t]\!],[\![\tau]\!])$. In this case $\sigma = \sigma_1\sigma_2 = \underbrace{\frac{l_1}{\chi_1}\cdots\frac{l_k}{\chi_k}}_{\text{occurrences in } [\![t]\!]} \quad \underbrace{\frac{l_{k+1}}{\chi_{k+1}}\cdots\frac{l_p}{\chi_p}}_{\text{occurrences in } [\![\tau]\!]}$.

  Then:

$$\begin{aligned}(\!| [\![@(t,\tau)]\!]\sigma |\!)' &= @((\!| [\![t]\!]\sigma_1 |\!)', (\!| [\![\tau]\!]\sigma_2 |\!)') \\ &= @(t\sigma_1, \tau\sigma_2) \\ &= @(t,\tau)\sigma\end{aligned}$$

  by induction on t and τ.

- $[\![u^*[\phi]]\!] = [\![u^*]\!] \{\!| \phi |\!\}$. In this case $\sigma = \sigma_1\sigma_2 = \underbrace{\frac{l_1}{\chi_1}\cdots\frac{l_k}{\chi_k}}_{\text{occurrences in } [\![u^*]\!]} \quad \underbrace{\frac{l_{k+1}}{\chi_{k+1}}\cdots\frac{l_p}{\chi_p}}_{\text{occurrences in } \{\!| \phi |\!\}}$.

  Then:

$$\begin{aligned}(\!| [\![u^*[\phi]]\!]\sigma |\!)' &= (\!| [\![u^*]\!]\sigma_1 |\!)'(\!| \{\!| \phi |\!\}\sigma_2 |\!)' \\ &= u^*[\phi]\sigma\end{aligned}$$

  by induction on u^* and ϕ.

where we extend $(\!|-|\!)'$ such that:

- $(\!| \{\!| \vec{\chi_p} \leftarrow \tau |\!\} |\!)' = \{(\!| [\![\tau]\!] |\!)'/\chi_p\}$,
- $(\!| \{\!| \vec{x_p} \twoheadleftarrow \mathcal{A}y.\langle t_1,\ldots,t_p \rangle \{\!| \Phi |\!\} |\!\} |\!)' = \{(\mathcal{A}y.(\!| [\![t_i]\!] \{\!| \Phi |\!\} |\!)')/x_i\}_{i\leq p}$,
- $(\!| \{\!| \Phi |\!\} |\!)' = (\!| \{\!| \phi_1 |\!\} |\!)' \ldots (\!| \{\!| \phi_n |\!\} |\!)'$,
- $\{\!| \Phi |\!\} = \{\!| \phi_1 |\!\} \ldots \{\!| \phi_n |\!\}$ where $[\Phi] = [\phi_1]\ldots[\phi_n]$

Finally, since $\{\!| \Phi |\!\}$ and σ are inverse operations, we have

$$\begin{aligned}(\!| [\![u^*]\!] |\!) &= (\!| [\![u^*]\!]\sigma |\!)'\Phi \\ &= u^*\sigma\Phi \\ &= u^*\end{aligned}$$

□

Lemma 6.2.3 (Atomic reduction simulates $\Lambda\mu S$-reduction). If $T \longrightarrow_{\beta,\mu} U$, then we have $(\!| T |\!) \longrightarrow^+ (\!| U |\!)$.

Proof. By induction on T. Recall that $\frac{l_i}{\chi_i}$ means replacing χ_i with l_i fresh distinct variables. Let $\sigma = \underbrace{\frac{l_1}{\chi_1} \dots \frac{l_p}{\chi_p}}_{\text{occurrences in } U}$, replacing the different occurrences of each free variable χ_i with fresh, distinct variables. Let $\Phi = [\vec{\chi_1}^{l_1} \leftarrow \chi_1] \dots [\vec{\chi_p}^{l_p} \leftarrow \chi_p]$. We use the induction hypothesis: if $T \longrightarrow_{\beta,\mu} U$, then $(\!|\, T\sigma \,|\!)' \longrightarrow_{\beta,\mu} (\!|\, U\sigma \,|\!)'$.

- If T is a redex:
  - If $T = (\lambda x.U)V \longrightarrow_\beta U\{V/x\}$, then (suppose $|U|_x \neq 1$)

$$\begin{aligned}(\!|\, T \,|\!) &= (\lambda x.(\!|\, U\sigma_1 \,|\!)'[\vec{x_p} \leftarrow x])(\!|\, V\sigma_2 \,|\!)'\Phi \\ &\longrightarrow_\beta (\!|\, U\sigma_1 \,|\!)'[\vec{x_p} \leftarrow (\!|\, V\sigma_2 \,|\!)']\Phi \\ &= (\!|\, U\{V/x\} \,|\!)\end{aligned}$$

  - Similar for the other redexes.
- If $T = @(U, \tau) \longrightarrow_{\beta,\mu} @(U', \tau)$, then

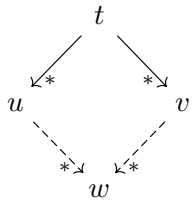
$$\begin{aligned}(\!|\, T \,|\!) &= @((\!|\, U\sigma_1 \,|\!), (\!|\, \tau\sigma_2 \,|\!))\Phi \\ &\longrightarrow^+ @((\!|\, U'\sigma_1 \,|\!)', (\!|\, \tau \,|\!)'\sigma_2)\Phi \\ &= (\!|\, @(U', \tau)\sigma \,|\!)'\Phi \\ &= (\!|\, @(U', \tau) \,|\!)\end{aligned}$$

- If $T = \mathcal{A}x.U \longrightarrow_{\beta,\mu} \mathcal{A}x.U'$, then (suppose $|U|_x \neq 1$)

$$\begin{aligned}(\!|\, T \,|\!) &= (\mathcal{A}x.(\!|\, U\sigma \,|\!)'[\vec{x_p} \leftarrow x])\Phi \\ &\longrightarrow^+ (\mathcal{A}x.(\!|\, U'\sigma \,|\!)'[\vec{x_p} \leftarrow x])\Phi \\ &= (\!|\, (\mathcal{A}x.U')\sigma \,|\!)'\Phi \\ &= (\!|\, \mathcal{A}x.U' \,|\!)\end{aligned}$$

□

Theorem 6.2.4 (Confluence of $\Lambda\mu S_a$). Let $t, u, v \in \Lambda\mu S_a$. If $t \longrightarrow^*_{\beta,\mu,s} u$ and $t \longrightarrow^*_{\beta,\mu,s} v$, there exists $w \in \Lambda\mu S_a$ such that $u, v \longrightarrow^*_{\beta,\mu,s} w$.



Proof. Suppose $t \longrightarrow^* u$ and $t \longrightarrow^* v$. Then in $\Lambda\mu S$ we have $[\![\, t \,]\!] \longrightarrow^* [\![\, u \,]\!]$ and $[\![\, t \,]\!] \longrightarrow^* [\![\, v \,]\!]$. By confluence of $\Lambda\mu S$, there exists $W \in \Lambda\mu S$ such that: $[\![\, u \,]\!] \longrightarrow^* W$ and $[\![\, v \,]\!] \longrightarrow^* W$. Let u_0, v_0 be the sharing normal forms of u, v. Then we have $(\!|\, [\![\, u \,]\!] \,|\!) = u_0$ and $(\!|\, [\![\, v \,]\!] \,|\!) = v_0$. Then, applying $(\!|-|\!)$ to $[\![\, u \,]\!]$ gives: $u_0 \longrightarrow^* (\!|\, W \,|\!)$ and $v_0 \longrightarrow^* (\!|\, W \,|\!)$. Combined, we get confluence, i.e. $u, v \longrightarrow^* (\!|\, W \,|\!)$. □

Chapter 7

Fully-lazy sharing

The atomic $\lambda\mu$-calculus satisfies fully-lazy sharing (in the sense of Wadsworth [Wad71]), which restricts duplication of subexpressions to *skeletons*, the remaining *maximal free subexpressions* (or *constants*) staying shared and being lifted outside the scope of the closure instead. The maximal free subexpression corresponds to the largest portion of a term t that can be shared and eventually only evaluated once, leaving the rest (skeletons) to be duplicated.

Let $\mathcal{V}$ be a set of (λ and μ) variables. Let $t^*, s^* \in \Lambda\mu S_a$. A *$\mathcal{V}$-free subexpression* s^* of t^* is a subexpression of t^* such that $FV(s^*) \subseteq FV(t^*)\backslash\mathcal{V}$. A $\mathcal{V}$-free subexpression is *maximal* when for any $\mathcal{V}$-free subexpression r^* of t^*, s^* is not a subexpression of r^*. A *free subexpression* s^* of t^* is a $\emptyset$-free subexpression of t^*. Intuitively, a free subexpression s^* of t^* is such that $t^* = u^*\{s^*/\chi\}$ for some u^*, the substitution not capturing any variable. It follows that the duplication $[\chi'_1, \ldots, \chi'_p \leftarrow t^*]$ becomes $\{u_1/\chi'_1\}\ldots\{u_p/\chi'_p\}[\chi_1, \ldots, \chi_p \leftarrow s^*]$, where u_i are *variants* of u that will be defined later. A maximal free subexpression s^* of t^* is the biggest subexpression of t^* such that if a bounded variable x of t^* appears in s^*, then the binder $\mathcal{A}x$ also appears in s^*.

Definition 7.0.1. A *maximal free subexpression* s^* of $\lambda x.t$ (resp. $\mu\alpha.t$) is a maximal x-free (resp. α-free) subexpression of t^*.

The intuition is that if we want to duplicate $\lambda x.t$ (resp. $\mu\alpha.t$), we can push the "constant" expressions (corresponding to maximal free subexpressions) towards the outside. The remaining parts, including the bound variable x (resp. α), will be duplicated.

Definition 7.0.2. A *fresh variant* t^{*i} of an expression t^* replaces occurrences of variables χ of t^* by fresh variables χ^i.

Definition 7.0.3.

- Let $u_1^*, \ldots, u_k^*$ be the maximal free subexpressions of an abstraction $\mathcal{A}x.t$. The *skeleton of an abstraction* $\mathcal{A}x.t$ is a term $\mathcal{A}x.t'$, where t' is obtained from t by replacing u_i^* by fresh variables χ_i (or α_j). Thus $\mathcal{A}x.t = (\mathcal{A}x.t')\{u_i^*/\chi_i\}_{i\leq k}$.

- Let $\chi, \vec{\chi_p}$ be variables in $\mathbb{T} \cup \mathbb{S}$, and let $t^*, u^* \in \mathbb{T} \cup \mathbb{S}$ be terms or streams. The $\mathcal{V}$-*skeleton* $skel_{\mathcal{V}}(t^*)$ of a basic expression $t^\star$, where $\mathcal{V}$ is a set of variables such that none is bound in $t^\star$ is:

$$skel_{\mathcal{V}}(\chi) = \chi$$

$$skel_{\mathcal{V}}(\mathcal{A}x.t) = \mathcal{A}x.(skel_{\mathcal{V}\cup\{x\}}(t)[x^1, \ldots, x^k \leftarrow x])$$

$$skel_{\mathcal{V}}(@(t, u^*)) = \begin{cases} \chi' & : \text{ if } FV(@(t,u^*)) \cap \mathcal{V} = \emptyset \\ @(skel_{\mathcal{V}}(t), \chi') & : \text{ if } FV(u^*) \cap \mathcal{V} = \emptyset \\ @(y, skel_{\mathcal{V}}(u^*)) & : \text{ if } FV(t) \cap \mathcal{V} = \emptyset \\ @(skel_{\mathcal{V}}(t), skel_{\mathcal{V}}(u^*)) & : \text{ otherwise} \end{cases}$$

$$skel_{\mathcal{V}}(t^*[\vec{\chi_p} \leftarrow u^*]) = \begin{cases} \chi' & : \text{ if } FV(t^*[\vec{\chi_p} \leftarrow u^*]) \cap \mathcal{V} = \emptyset \\ skel_{\mathcal{V}}(t^*) & : \text{ if } FV(u^*) \cap \mathcal{V} = \emptyset \\ (skel_{\mathcal{V}\cup\{\vec{\chi_p}\}}(t^*))\sigma & : \text{ otherwise} \end{cases}$$

  where y, χ' are fresh variables, and the variants x^i come from the substitutions σ. The substitution $\sigma = \{(skel_{\mathcal{V}}(u^*))^i/\chi_i\}_{i\leq p}$.

- The *skeleton* $skel(t^*)$ of an expression t^* is $skel_\emptyset(t^*)$.

We now show the core lemma for full-laziness:

Lemma 7.0.4. Let $\mathcal{V}$ be a set of variables, let $t^*, u^* \in \Lambda\mu S_a^-$ be terms or streams, let χ_i, χ'_j be variables.

An expression $u^*[\vec{\chi_p} \leftarrow t^*]$ can be reduced to $u^*\{skel_{\mathcal{V}}(t^*)^i/\chi_i\}_{i\leq p}[\Gamma][\Delta]$ where Γ gathers all the sharings of the form $[\vec{\chi'_k} \leftarrow \chi']$ for each $\chi' \in FV(t^*) \cap \mathcal{V}$.

Proof. By induction on t. Let $v^*, w^* \in \mathbb{T} \cup \mathbb{S}$ be terms or streams, let χ, χ_k, χ'_l be variables in $\mathbb{T} \cup \mathbb{S}$.

- Let $t = \chi$. Then

$$\begin{aligned} u^*[\vec{\chi_p} \leftarrow \chi] &=_\alpha u^*\{\chi^i/\chi_i\}_{i\leq p}[\vec{\chi^p} \leftarrow \chi] \\ &= u^*\{skel_{\mathcal{V}}(\chi)^i/\chi_i\}_{i\leq p}[\vec{\chi^p} \leftarrow \chi] \end{aligned}$$

- If $t = \mathcal{A}x.v$, let $s = skel_{\mathcal{V}\cup\{x\}}(v)$. Then

$$\begin{aligned} u^*[\vec{x_p} \leftarrow \mathcal{A}x.v] &\longrightarrow_s u^*[\vec{x_p} \nleftarrow \mathcal{A}x.\langle\, \vec{y_p} \,\rangle[\vec{y_p} \leftarrow v]] \\ &\longrightarrow_s^* u^*[\vec{x_p} \nleftarrow \mathcal{A}x.\langle\, \vec{s^p} \,\rangle[\vec{z_k} \leftarrow x][\Gamma][\Delta]] \\ &\text{(by induction hypothesis)} \\ &\longrightarrow_s^* u^*[\vec{x_p} \nleftarrow \mathcal{A}x.\langle\, \vec{s^p} \,\rangle[\vec{z_k} \leftarrow x]][\Gamma][\Delta] \\ &\longrightarrow_s u^*\{\mathcal{A}x^i.s^i[\vec{z_l^i} \leftarrow x^i]/x_i\}_{i\le p}[\Gamma][\Delta] \\ &\text{(by distributor elimination)} \\ &= u^*\{(skel_{\mathcal{V}}(\lambda y.v))^i/x_i\}_{i\le p}[\Gamma][\Delta] \end{aligned}$$

  where z_k and z_l are as in the distributor elimination rule.

- Let $t = @(v, w^*)$.

  - Suppose $FV(v) \cap \mathcal{V} \neq \emptyset$ and $FV(w^*) \cap \mathcal{V} = \emptyset$. Then:

$$\begin{aligned} u^*[\vec{\chi_p} \leftarrow @(v, w^*)] &\longrightarrow_s u^*\{@(y_i, \chi_i')/\chi_i\}_{i\le p}[\vec{y_p} \leftarrow v][\vec{\chi_p'} \leftarrow w^*] \\ &\longrightarrow_s^* u^*\{@(y_i, \chi_i')/\chi_i\}_{i\le p}\{skel_{\mathcal{V}}(v)^i/y_i\}_{i\le p} \\ &\{skel_{\mathcal{V}}(w^*)^i/\chi_i'\}_{i\le p}[\Gamma][\Delta] \\ &\text{(by induction hypothesis)} \\ &= u^*\{@(skel_{\mathcal{V}}(v)^i, skel_{\mathcal{V}}(w^*)^i)/\chi_i\}_{i\le p} \\ &[\Gamma][\Delta] \end{aligned}$$

  - The three remaining cases are proved similarly.

- Let $t = w^*[\vec{\chi_m'} \leftarrow v^*]$. Let $\sigma = \{(skel_{\mathcal{V}}(v^*))^j/\chi_j'\}_{j\le m}$.

– Suppose $FV(w^*) \cap \mathcal{V} \neq \emptyset$ and $FV(v^*) \cap \mathcal{V} \neq \emptyset$. Then:

$$
\begin{aligned}
u^*[\vec{\chi_p} \leftarrow w^*[\vec{\chi'_m} \leftarrow v^*]] &\longrightarrow_s u^*[\vec{\chi_p} \leftarrow w^*][\vec{\chi'_m} \leftarrow v^*] \\
&\longrightarrow_s^* u^*\{(skel_{\mathcal{V} \cup \{\vec{\chi'_m}\}}(w^*))^i/\chi_i\}_{i \leq p}[\vec{\chi'_{1^{k_1}}} \leftarrow \chi'_1] \ldots \\
&[\vec{\chi'_{m^{k_m}}} \leftarrow \chi'_m][\Gamma_1][\Delta_1][\vec{\chi'_m} \leftarrow v^*] \\
&\text{(by induction hypothesis)} \\
&\longrightarrow_s^* u^*\{(skel_{\mathcal{V} \cup \{\vec{\chi'_m}\}}(w^*))^i/\chi_i\}_{i \leq p}[\vec{\chi'_{1^{k_1}}} \leftarrow (skel_{\mathcal{V}}(v^*))^1] \\
&\ldots [\vec{\chi'_{m^{k_m}}} \leftarrow (skel_{\mathcal{V}}(v^*))^m][\Gamma_1][\Delta_1][\Gamma_2][\Delta_2] \\
&\longrightarrow_s^* u^*\{(skel_{\mathcal{V} \cup \{\vec{\chi'_m}\}}(w^*)\sigma)^i/\chi_i\}_{i \leq p}[\Gamma_3][\Delta_3] \\
&[\Gamma_1][\Delta_1][\Gamma_2][\Delta_2] \\
&= u^*\{(skel_{\mathcal{V}}(w^*[\vec{\chi'_m} \leftarrow v^*]))^i/\chi_i\}_{i \leq p}[\Gamma][\Delta]
\end{aligned}
$$

where $[\Gamma] = [\Gamma_1][\Gamma_2][\Gamma_3]$ and $[\Delta] = [\Delta_1][\Delta_2][\Delta_3]$. We use the fact that $skel_{\mathcal{V}}$ is idempotent.

– The other cases are shown similarly.

□

Note that Δ gathers the maximal free subexpressions, which then remain shared, whereas skeletons are duplicated. This proof shows a fully-lazy strategy, in which basic expressions (inside closures) get reduced to basic expressions, such that introduced distributors are eliminated. The strategy is to first push closures outside of the expression before performing duplications. Non-basic expressions (distributors) appear when reducing basic expressions, so we can start from basic expressions (and eventually we remove distributors).

From this lemma, we can conclude:

Theorem 7.0.5. The atomic $\lambda\mu$-calculus satisfies fully-lazy sharing.

Chapter 8

Conclusions

In this section, we discuss the more general conclusions and perspectives to be drawn from our work. We give a brief summary of our results, then we address some possible future perspectives.

8.1 Results

In this thesis we have combined the $\lambda\mu$-calculus [Par92] and the atomic λ-calculus [GHP13] to construct the atomic $\lambda\mu$-calculus. For explicit sharings to behave well during μ-reduction, we work on a variant of the calculus using streams. As for the atomic λ-calculus, since atomic duplications correspond to the medial rule which bears no resemblance to any sequent calculus rule, deep inference methodology is the most natural way to describe terms and reductions. Therefore we use the most general formalism, open deduction, to type terms of the calculus. Following the correspondence of the $\lambda\mu$-calculus with classical logic, we first attempt to build a multiple conclusion system, which unfortunately leads to many seemingly superfluous steps and creates additional bureaucracy between disjunctions and meta-disjunctions to distinguish the main formula in a conclusion. Observing the symmetry between λ and μ, and the fact that multiple conclusions come with μ-variables that are not yet bound, we retrieve a single-conclusion system by "hiding" those variables until they get bound, and obtain a μ-abstraction rule very similar to that for λ-abstraction. We finally show that types are preserved under reduction.

Our main result for this calculus is preservation of strong normalization, ensuring that $\Lambda\mu S$-terms that always terminate remain that way in the atomic $\lambda\mu$-calculus. In particular PSN would have been shown if a reduction step in the atomic calculus

coincided with at least a reduction step in the $\Lambda\mu S$-calculus, but this is not true for two reasons. The first reason has to do with sharing reductions, which correspond to zero steps in the $\Lambda\mu S$-calculus. This is not problematic since sharing reduction is strongly normalizing, therefore infinite paths must come from λ or μ-reductions. The proof for the strong normalization of sharing reductions differs from the atomic λ-calculus proof in [GHP13], being more faithful to the semantics i.e. graphs mapping onto terms. The idea is to construct a strictly decreasing measure, keeping track of weakening reductions, the number of duplications of subterms, the number of closures, and the lengths to reach closures. In particular weakening reductions are strongly normalizing.

The second reason is that infinite reductions can happen inside weakenings of atomic terms, which are then discarded in their interpretation in the $\Lambda\mu S$-calculus. We need to show that starting from a $\Lambda\mu S$-term we cannot fall into that situation, i.e. if we get an infinite reduction inside a weakening in the atomic calculus, this infinite reduction could have remained outside the weakening all along, and its counterpart in the $\Lambda\mu S$-calculus would have had an infinite path as well. To do that we introduce an intermediate calculus, the $\Lambda\mu S$-calculus with explicit weakenings, and split PSN into two parts, PSN between the weakening calculus and the atomic calculus, and PSN between the $\Lambda\mu S$-calculus and the weakening calculus. The way we designed the weakening calculus is such that the former part is satisfied. The latter part requires defining an exhaustive reduction strategy ensuring that an infinite path in the weakening calculus is found should it exist, and stays infinite when translated back to the $\Lambda\mu S$-calculus. Our strategy is more general than the presentation from [GHP13], and helps understand better the properties of the weakening calculus. Therefore whenever a $\Lambda\mu S$-term is strongly normalizing, we show that the perpetual strategy of its translation in the weakening calculus terminates, making the weakening term strongly normalizing as well.

Other results we show are confluence and full-laziness, making the atomic $\lambda\mu$-calculus an efficient model for programming languages.

8.2 Next steps

Regarding ways to build upon our work, there are three possible directions. Firstly, we could apply our techniques to investigate the implementation of a wider range of side-effects and handlers, as with algebraic effects [FS14]. Secondly, concerning atomic calculi, further work is in the direction of capturing optimality by considering the two further medial rules of intuitionistic deep inference. Lastly, we could move towards

implementation. A efficient implementation is currently studied through Sherratt's directed atomic λ-calculus. Sharing reductions are not *local*, since some rules can only be applied on subterms after inspecting larger subterms (typically to check whether a variable is free or bound), which makes implementation more complicated. By using director strings [KS88] and therefore keep track of free variables, the directed atomic λ-calculus aims to give an efficient [SFM03] (i.e. keeping full laziness) implementation of the atomic λ-calculus.

Bibliography

[ABM14] Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines (long version). *CoRR*, abs/1406.2370, 2014.

[Acc12] Beniamino Accattoli. An abstract factorization theorem for explicit substitutions. In *RTA*, volume 15 of *LIPIcs*, pages 6–21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[ACCL91] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *J. Funct. Program.*, 1(4):375–416, 1991.

[AG98] Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages*. Cambridge University Press, 1998.

[Bal12] Thibaut Balabonski. A unified approach to fully lazy sharing. *SIGPLAN Not.*, 47(1):469–480, January 2012.

[Bar84] Henk Barendregt. *The Lambda Calculus Its Syntax and Semantics*, volume 103. North Holland, revised edition, 1984. http://www.cs.ru.nl/ henk/Personal Webpage.

[BGGP15] Paola Bruscoli, Alessio Guglielmi, Tom Gundersen, and Michel Parigot. Quasipolynomial normalisation in deep inference via atomic flows and threshold formulae. *Logical Methods in Computer Science*, 2015. To appear.

[BL05] Kai Brünnler and Stéphane Lengrand. On two forms of bureaucracy in derivations. In Paola Bruscoli, François Lamarche, and Charles Stewart, editors, *Structures and Deduction*, pages 69–80. Technische Universität Dresden, 2005. ICALP Workshop. ISSN 1430-211X.

[BM08] Kai Brünnler and Richard McKinley. An algorithmic interpretation of a deep inference system. In Iliano Cervesato, Helmut Veith, and Andrei

Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 5330 of *Lecture Notes in Computer Science*, pages 482—496. Springer-Verlag, 2008.

[Brü03a] Kai Brünnler. Atomic cut elimination for classical logic. In M. Baaz and J. A. Makowsky, editors, *Computer Science Logic (CSL)*, volume 2803 of *Lecture Notes in Computer Science*, pages 86–97. Springer-Verlag, 2003.

[Brü03b] Kai Brünnler. Two restrictions on contraction. *Logic Journal of the IGPL*, 11(5):525–529, 2003.

[Brü06] Kai Brünnler. Cut elimination inside a deep inference system for classical predicate logic. *Studia Logica*, 82(1):51–71, 2006.

[BT01] Kai Brünnler and Alwen Fernanto Tiu. A local system for classical logic. In R. Nieuwenhuis and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 2250 of *Lecture Notes in Computer Science*, pages 347–361. Springer-Verlag, 2001.

[CH00] Pierre-Louis Curien and Hugo Herbelin. The duality of computation. In Martin Odersky and Philip Wadler, editors, *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000.*, pages 233–243. ACM, 2000.

[Chu32] Alonzo Church. A set of postulates for the foundation of logic part i. *Annals of Mathematics*, 33(2):346–366, 1932. http://www.jstor.org/stable/1968702Electronic Edition.

[Cur34] H. Curry. Functionality in combinatorial logic. In *Proceedings of National Academy of Sciences*, volume 20, pages 584–590, 1934.

[Das12] Anupam Das. Complexity of deep inference via atomic flows. In S. Barry Cooper, Anuj Dawar, and Benedikt Löwe, editors, *Computability in Europe*, volume 7318 of *Lecture Notes in Computer Science*, pages 139–150. Springer-Verlag, 2012.

[DP01] René David and Walter Py. $\lambda\mu$-calculus and Böhm's theorem. *Journal of Symbolic Logic*, 2001.

[FS14] Marcelo P. Fiore and Sam Staton. Substitution, jumps, and algebraic effects. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of*

the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014, pages 41:1–41:10. ACM, 2014.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen I, II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934–1935. English translation in: The Collected Papers of Gerhard Gentzen, M.E. Szabo (ed.), North-Holland 1969.

[Gen69] Gerhard Gentzen. Untersuchungen über das logische Schließen I, II. In M E Szabo, editor, *The Collected Papers of Gerhard Gentzen*. North-Holland Publ. Co., Amsterdam, 1969.

[GHP13] Tom Gundersen, Willem Heijltjes, and Michel Parigot. Atomic lambda calculus: A typed lambda-calculus with explicit sharing. In Orna Kupferman, editor, *28th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 311–320. IEEE, 2013.

[Gir72] Jean-Yves Girard. Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur. 1972.

[Gir91] Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science*, 1(3):255–296, 1991.

[Gri90] Timothy G. Griffin. A formulae-as-types notion of control. In *In Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 47–58. ACM Press, 1990.

[GS10] Marco Gaboardi and Alexis Saurin. A foundational calculus for computing with streams. In *ICTCS*, 2010.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. New York, NY, USA, 1989.

[Gug07] Alessio Guglielmi. A system of interaction and structure. *ACM Transactions on Computational Logic*, 8(1):1:1–64, 2007.

[How80] William Howard. The formulae-as-types notion of construction. In Haskell Curry, Jonathan Seldin, and Roger Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[HS97] Martin Hofmann and Thomas Streicher. Continuation models are universal for lambda-mu-calculus. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 387–395. IEEE Computer Society, 1997.

[JL82] Jean-Pierre Jouannaud and Pierre Lescanne. On multiset orderings. 15:57–63, 09 1982.

[KR09] Delia Kesner and Fabien Renaud. The prismoid of resources. In Rastislav Královic and Damian Niwinski, editors, *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, volume 5734 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 2009.

[KS88] Richard Kennaway and Ronan Sleep. Director strings as combinators. *ACM Trans. Program. Lang. Syst.*, 10(4):602–626, October 1988.

[Lam90] John Lamping. An algorithm for optimal lambda calculus reduction. In *POPL*, pages 16–30, 1990.

[Len06] Stéphane Lengrand. *Normalisation & Equivalence in Proof Theory & Type Theory*. PhD thesis, Université Paris 7 & University of St Andrews, 2006.

[Lév80] Jean-Jacques Lévy. Optimal reductions in the lambda-calculus. In *To H.B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[Mel95] Paul-André Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In *Proceedings of the Second International Conference on Typed Lambda Calculi and Applications*, TLCA '95, pages 328–334, London, UK, UK, 1995. Springer-Verlag.

[Ong96] C.-H. Luke Ong. A semantic view of classical proofs: Type-theoretic, categorical, and denotational characterizations (preliminary extended abstract). In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 230–241. IEEE Computer Society, 1996.

[Par92] Michel Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *LPAR*, pages 190–201, 1992.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda-calculus. *Theor. Comput. Sci.*, 1(2):125–159, 1975.

[Sau08] Alexis Saurin. *Une étude logique du contrôle*. Thèse de doctorat, École Polytechnique, 2008.

[Sel01] Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.

[SFM03] François-Régis Sinot, Maribel Fernández, and Ian Mackie. Efficient reductions with director strings. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications, 14th International Conference, RTA 2003, Valencia, Spain, June 9-11, 2003, Proceedings*, volume 2706 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2003.

[SS05] Charles Stewart and Phiniki Stouppa. A systematic proof theory for several modal logics. In Renate Schmidt, Ian Pratt-Hartmann, Mark Reynolds, and Heinrich Wansing, editors, *Advances in Modal Logic (AiML)*, volume 5, pages 309–333. King's College Publications, 2005.

[Str02] Lutz Straßburger. A local system for linear logic. Technical Report WV-02-01, Technische Universität Dresden, 2002.

[Tiu06] Alwen Fernanto Tiu. A local system for intuitionistic logic. In Miki Hermann and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 4246 of *Lecture Notes in Computer Science*, pages 242–256. Springer-Verlag, 2006.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, 2013.

[Wad71] Christopher P. Wadsworth. *Semantics and Pragmatics of the Lambda-Calculus*. Ph. D. dissertation, Oxford University, Oxford, England, September 1971.